| ANNEE | NIVEAU | SUJET | DESCRIPTIF | LANGAGE | RESPONSABLE | DIFFICULTE | GROUPE |
|---|---|---|---|---|---|---|---|
| 2009-2010 | L3 | ELIZA en français | On vous propose d'implémenter une version française de ELIZA. ELIZA est historiquement le premier succès spectaculaire du TAL. Il s'agit d'un petit agent conversationnel qui simule une conversation entre vous et un psychanalyste de patience infinie. Exemple de conversation en français :<br>Système : "Comment allez-vous ?"<br>Utilisateur : "Je suis déprimé."<br>S : "Dites-moi pourquoi vous êtes déprimé."<br>U : "Je passe trop de temps avec les ordinateurs."<br>S : "Pourquoi mentionnez-vous les ordinateurs ?"<br>U : "Parce que tu en es un."<br>S : "Quels sont vos rapports avec les ordinateurs ?"<br>U : "Difficiles."<br>S : etc.<br>Ce petit agent fonctionne principalement à l'aide de techniques élémentaires de pattern matching (expressions régulières) et vous demande de manipuler une pile simulant sa mémoire. Le sujet se focalisera sur les points suivants :<br>• Identifier un mot clé et axer la réponse sur ce mot clé (exemple : retenir un groupe de deux mots commençant par un déterminant, "les ordinateurs" dans "Je passe trop de temps avec les ordinateurs"),<br>• Revenir à un mot clé quand l'énoncé courant ne permet de générer aucune réponse,<br>• Exploiter des phrases à trou (exemple : "Pourquoi mentionnez-vous" + mot clé),<br>• Traduire les mots référant aux interlocuteurs : "je", "vous", "moi", "me", etc. (exemple : "je suis déprimé" devient "vous êtes déprimé" dans "Dites-moi pourquoi vous êtes déprimé").<br>• Sortir des réponses types au hasard (exemple : "Comment allez-vous ?"). | Java | Grégoire Winterstein | Difficile | 2 personnes |
| | | Détection de la langue d'un texte | Pour détecter la langue d'un texte, on peut constituer une base de connaissances à partir d'un corpus de textes classés par langue. Pour chaque langue, un premier programme (à écrire) recueillera des statistiques significatives, basées sur les lettres (par exemple, il y a plus de "w" en anglais qu'en français). Le choix du modèle probabiliste employé et de ses paramètres (bigrammes, trigrammes) devra être justifié dans le rapport. On peut en proposer plusieurs et discuter de leurs avantages et inconvenients (rapport entre précision de la reconnaissance et volume de la base de connaissances ou longueur du texte nécessaire pour reconnaître sa langue).<br>Un deuxième programme (à écrire), utilisera ces bases de connaissances pour reconnaître la langue d'un texte.<br>Vous utiliserez un corpus d'apprentissage comprenant des textes plus ou moins variés d'un certain nombre de langues (le plus de langues possible, au moins 4). Le corpus de test ne devra pas contenir de texte appartenant au corpus d'apprentissage.<br>Amélioration possible : gérer des textes dans différents encodages, pour les langues à alphabet non latin. | Java | Grégoire Winterstein | Facile à moyen | 2 personnes |
| | | Correcteur orthographique | On se propose de réaliser un correcteur orthographique (lexical), qui, disposant d'un dictionnaire de formes fléchies, détecte les mots mal orthographiés, et propose si possible une correction.<br>Le programme prend un texte (ASCII brut) en entrée, et pour chaque forme non présente dans le dictionnaire, propose à l'utilisateur :<br>• de choisir un de ces remplaçants<br>o pour l'occurrence, ou pour toutes les occurrences<br>• de fournir un remplaçant<br>o pour l'occurrence, ou pour toutes les occurrences<br>o si le remplaçant n'est pas dans le lexique, | Java | Grégoire Winterstein | | 2 personnes |

| ANNEE | NIVEAU | SUJET | DESCRIPTIF | LANGAGE | RESPONSABLE | DIFFICULTE | GROUPE |
|---|---|---|---|---|---|---|---|
| | | | ☐ de l'insérer dans le lexique pour ce texte,<br>☐ de l'insérer dans le lexique stable<br>• d'ignorer la correction, c'est-à-dire conserver le mot initial, pour cette occurrence ou pour toutes, en l'insérant ou non dans le lexique.<br>Les remplaçants proposés par le programme (premier cas plus haut) seront des mots figurant dans le dictionnaire, et ayant une certaine proximité avec le mot fautif.<br>Pour la recherche des mots proches, on appliquera diverses heuristiques, basées sur la forme du mot :<br>• repérage des bigrammes ou trigrammes impossibles en français<br>• redoublement ou dé-doublement de consonnes<br>• suppression/insertion de diacritiques<br>• distance d'édition<br>• etc<br>Ces heuristiques, qui peuvent être plus nombreuses, seront étudiées linguistiquement afin de déterminer précisément leurs conditions d'application. | | | | |
| | | Réaccentuation | Le programme est chargé de remettre les accents et autres signes diacritiques manquant dans un texte fourni en typographie dite pauvre.<br>On s'aidera d'un lexique de formes fléchies fourni. Bien sûr, certaines formes peuvent être réaccentuées directement, et d'autres sont ambigues. Une fois que l'algorithme principal sera établi, on envisagera des heuristiques pour lever les ambiguïtés. Exemple :<br>La ou le francais n'est pas accentue,<br>il y a de la gene,<br>mais quand le systeme m'accentue,<br>je suis moins gene! | Java | Grégoire Winterstein | | 2 personnes |
| | | Entités nommées | On regroupe sous le terme "entités nommées" les noms de personnes, de lieux, de dates, noms d'entreprises, adresses, etc. Il s'agit d'expressions qui dénotent une entité unique de façon presque indépendante du contexte. On s'intéresse aux entités nommées pour plusieurs raisons :<br>• elles constituent des syntagmes qui peuvent être relativement complexes au point de vue syntaxique (par exemple une adresse, ou un nom d'association) dont le repérage préalable peut grandement siplifier une analyse syntaxique ;<br>• dans une perspective de recherche d'information, la reconnaissance des entités nommées permet de savoir de quoi parle un texte ;<br>• elles sont nécessaires pour la résolution des anaphores.<br>Il s'agit dans ce projet de repérer de la façon la plus complète possible dans un texte étiqueté ou non, les entités de type "personne". Pour cela, on envisagera un algorithme en deux étapes (qui peuvent se répéter) :<br>• au moyen de règles générales et de dictionnaires spécialisés (noms propres, amorces --- c'est-à-dire mots qui introduisent systématiquement des entités nommées, comme 'Melle', etc.), constitution d'une "table des symboles" des entités présentes dans le texte ;<br>• à partir de cette table des symboles, et en tenant compte des formes variées sous lesquelles une même entités peut être désignée, recherche de nouvelles entités, voire de nouvelles règles trouvées précédemment.<br>L'idée est que le programme s'enrichit au fur et à mesure qu'il est utilisé. | Java | Grégoire Winterstein | Facile à très difficile | 2 ou 3 personnes |
| | | Extraction de collocations en corpus | Les collocations sont des ensembles de deux ou plusieurs mots qui sont plus fréquemment coocurrents que la normale. Il s'agit de constructions qui ont une syntaxe classique mais une sémantique qui n'est pas complètement compositionnelle. | Java, python ou perl et initiation à R (facultatif) | Grégoire Winterstein | Moyen | 2 personnes |

| ANNEE | NIVEAU | SUJET | DESCRIPTIF | LANGAGE | RESPONSABLE | DIFFICULTE | GROUPE |
|---|---|---|---|---|---|---|---|
| | | | Par exemple on considère que thé fort est une collocation car ces deux mots apparaissent assez souvent en même temps et le sens de fort dans cette séquence est légèrement altéré : il ne s'agit pas d'une grande force physique mais plutôt d'une forte concentration d'un agent du thé.<br>Ce sujet propose de partir à la découverte de différentes méthodes connues de détection de collocations en corpus reposant sur des tests d'hypothèses statistiques (comme par exemple le test du Chi2 ou le test exact de Fisher) en utilisant des logiciels d'analyse de données appropriés. | | | | |
| | | Clavier téléphone (T9) | La plupart des téléphones associent à chaque touche un certain nombre de lettres, permettant ainsi de transmettre des messages. Du fait qu'un chiffre ne correspond pas à une seule lettre, une suite de chiffres peut être ambiguë (elle peut correspondre à plus d'un mot). La suite 7-6-8-7, par exemple, correspond aux mots pour, sous et pots. Le but du projet est de réaliser un programme qui prend en entrée une suite de chiffre et un dictionnaire de formes fléchies et qui propose en sortie la liste des mots du dictionnaire qui correspondent à la suite des chiffres. Ces mots seront classés selon une fréquence représentée dans le dictionnaire. On attachera un soin particulier au choix de la structure de données, et à l'algorithme, qui doit faire le calcul en un temps raisonnable. | Java, python ou perl | Grégoire Winterstein | Moyen | 2 personnes |
| | M1 | Marquage des antécédents pronominaux en corpus | Il s'agit de mettre en oeuvre un ou plusieurs algorithmes de résolution d'anaphore sur un texte analysé syntaxiquement (Le French Treebank du laboratoire LLF pourra être utilisé).<br>A partir des occurrences pronominales de 3e personne marquées pour le genre et le nombre de leurs antécédents, il s'agit de proposer un ou plusieurs candidats dans la même phrase (pour les relatifs et les réfléchis) ou non (pour les personnels).<br>Les mots antécédents seront repérés par leur numéro dans la phrase (ID) éventuellement préfixé par le numéro de la phrase. | Java ou python | Pascal Amsili, Pascal Denis | Facile à très difficile | 2 personnes |
| | | Construction automatique d'un thésaurus distributionnel du français | Un thésaurus distributionnel est un dictionnaire dont chaque entrée est associée à une liste "voisins" distributionnels (à savoir des mots qui apparaîssent fréquemment dans les mêmes contextes). En plus de constituer une ressource intéressante pour la linguistique de corpus, ce type de thésaurus se montre extrêmement utile (comme complément ou même comme substitut à une ressource statique telle que Wordnet) pour de nombreuses tâches de TAL (parsing syntaxique, désambiguïsation lexicale...).<br><br>La construction d'un thésaurus distributionnel comprend trois grandes étapes: (i) l'extraction et le prétraitement d'un corpus de grande taille (plusieurs millions de mots), (ii) l'identification des contextes associés à chaque mot (ces contextes sont représentés sous la formes de simples n-grammes ou de triplets de dépendances syntaxiques), et (iii) un calcul de similarité entre contextes (ce calcul se fait typiquement sur base d'une mesure d'informatiion mutuelle).<br><br>Ce projet mettra donc en oeuvre cette procédure pour la construction d'un thésaurus automatique du français. En particulier, on utilisera comme corpus de départ le corpus issu de la campagne PASSAGE: un corpus de 100M de mots déjà prétraité et analysé en dépendances syntaxiques par différents analyseurs.<br><br>Ce sujet demande des compétences solides en programmation. | Java ou python | Pascal Denis | Moyen à difficile | 2 personnes |
| | | Réseau sémantique à partir d'un dictionnaire | À partir de définitions provenant de dictionnaires électroniques, préalablement étiquetées, il s'agit de construire un « réseau sémantique ».<br>Pour chaque entrée (qui peut correspondre à un des sens d'une lexie), on repèrera les mots pleins qui participent à sa définition (il faut donc (1) distinguer les mots pleins des mots « outils », (2) repérer la définition proprement dite parmi l'ensemble des informations associées à une entrée (catégorie, exemples, synonymes...)), et on construira un réseau reliant l'entrée à tous ces mots pleins.<br>Ce réseau pourra être stocké sous forme de fichier Ascii, dans un format du genre de celui de WordNet. On pourra aussi, éventuellement, proposer une interface graphique permettant de visualiser graphiquement le réseau, voire de le modifier. | Java ou python | Pascal Amsili | Moyen à très difficile | 2 ou 3 personnes |

| ANNEE | NIVEAU | SUJET | DESCRIPTIF | LANGAGE | RESPONSABLE | DIFFICULTE | GROUPE |
|---|---|---|---|---|---|---|---|
| | | | Dans un deuxième temps, on réalisera un programme exploitant ce réseau pour désambiguïser les sens d'un mot en contexte. Principe : étant donnée une phrase contenant le mot concerné, on repère les mots pleins qui l'entourent (contexte), et on recherche ces mots dans le réseau. Alors le sens correspondant est celui qui est le plus proche (dans un sens qu'il faut précisément définir) des mots pleins de son contexte.<br><br>D'autres exploitations d'un tel réseau peuvent être envisagées. | | | | |
| | | Extraction de dépendances à partir du French Treebank | Il s'agit d'expliciter les dépendances syntaxiques contenues dans les arbres syntagmatiques du French Treebank du laboratoire LLF . L'intérêt du sujet est d'une part pédagogique : il permet une familiarisation avec le traitement de corpus syntaxiquement annoté. Le passage de la constituance à la dépendance est en soi un problème linguistiquement intéressant, qui requiert une bonne maîtrise des concepts syntaxiques fondamentaux (constituants / fonctions / tête). D'autre part, le résultat de la conversion (le corpus converti en dépendances), directement exploitable pour par exemple extraire un lexique syntaxique, ou pour des expériences de parsing probabiliste en dépendances.<br><br>La mise en œuvre consistera en :<br>• Etudier le corpus syntaxiquement annoté (avec par exemple l'outil tregexp). Un programme de chargement du corpus en structures python vous sera fourni.<br>• Dégager et appliquer des règles de propagation de tête dans les constituants<br>• Convertir les constituants en dépendances, si possible dans un format standard, comme celui de la campagne d'évaluation CoNLL 2006.<br>Vous procèderez itérativement en étudiant les erreurs / échecs à chaque itération. | Python | Marie Candito | Moyen | 2 personnes |
| 2008-2009 | L3 | ELIZA en français | On vous propose d'implémenter une version française de ELIZA. ELIZA est historiquement le premier succès spectaculaire du TAL. Il s'agit d'un petit agent conversationnel qui simule une conversation entre vous et un psychanalyste de patience infinie. Exemple de conversation en français :<br>Système : "Comment allez vous ?"<br>Utilisateur : "Je suis déprimé."<br>S : "Dites-moi pourquoi vous êtes déprimé."<br>U : "Je passe trop de temps avec les ordinateurs."<br>S : "Pourquoi mentionnez-vous les ordinateurs ?"<br>U : "Parce que tu en es un."<br>S : "Quels sont vos rapports avec les ordinateurs ?"<br>U : "Difficiles."<br>S : etc<br>Ce petit agent fonctionne principalement à l'aide de techniques élémentaires de pattern matching (expressions régulières) et vous demande de manipuler une pile simulant sa mémoire. Le sujet se focalisera sur les points suivants :<br>Identifier un mot clé et axer la réponse sur ce mot clé (exemple : retenir un groupe de deux mots commençant par un déterminant, "les ordinateurs" dans "Je passe trop de temps avec les ordinateurs"),<br>Revenir à un mot clé quand l'énoncé courant ne permet de générer aucune réponse,<br>Exploiter des phrases à trou (exemple : "Pourquoi mentionnez-vous" + mot clé),<br>Traduire les mots référant aux interlocuteurs : "je", "vous", "moi", "me", etc. (exemple : "je suis déprimé" devient "vous êtes déprimé" dans "Dites-moi pourquoi vous êtes déprimé").<br>Sortir des réponses types au hasard (exemple : "Comment allez-vous ?"). | Java | Benoît Crabbé | Difficile | 2 personnes |

| ANNEE | NIVEAU | SUJET | DESCRIPTIF | LANGAGE | RESPONSABLE | DIFFICULTE | GROUPE |
|---|---|---|---|---|---|---|---|
| | | Détection de la langue d'un texte | Pour détecter la langue d'un texte, on peut constituer une base de connaissances à partir d'un corpus de textes classés par langue. Pour chaque langue, un premier programme (à écrire) recueillera des statistiques significatives, basées sur les lettres (par exemple, il y a plus de "w" en anglais qu'en français). Le choix du modèle probabiliste employé et de ses paramètres (bigrammes, trigrammes) devra être justifié dans le rapport. On peut en proposer plusieurs et discuter de leurs avantages et inconvenients (rapport entre précision de la reconnaissance et volume de la base de connaissances ou longueur du texte nécessaire pour reconnaître sa langue).<br><br>Un deuxième programme (à écrire), utilisera ces bases de connaissances pour reconnaître la langue d'un texte.<br><br>Vous utiliserez un corpus d'apprentissage comprenant des textes plus ou moins variés d'un certain nombre de langues (le plus de langues possible, au moins 4). Le corpus de test ne devra pas contenir de texte appartenant au corpus d'apprentissage.<br><br>Amélioration possible : gérer des textes dans différents encodages, pour les langues à alphabet non latin. | Java | Marie Candito | Facile à moyen | 2 personnes |
| | | Correcteur orthographique (approche graphémique) | On se propose de réaliser un correcteur orthographique (lexical), qui, disposant d'un dictionnaire de formes fléchies, détecte les mots mal orthographiés, et propose si possible une correction.<br>Le programme prend un texte (ASCII brut) en entrée, et pour chaque forme non présente dans le dictionnaire, propose à l'utilisateur :<br>de choisir un de ces remplaçants<br>pour l'occurrence, ou pour toutes les occurrences<br>de fournir un remplaçant<br>pour l'occurrence, ou pour toutes les occurrences<br>si le remplaçant n'est pas dans le lexique,<br>de l'insérer dans le lexique pour ce texte,<br>de l'insérer dans le lexique stable<br>d'ignorer la correction, c'est-à-dire conserver le mot initial, pour cette occurrence ou pour toutes, en l'insérant ou non dans le lexique.<br>Les remplaçants proposés par le programme (premier cas plus haut) seront des mots figurant dans le dictionnaire, et ayant une certaine proximité avec le mot fautif.<br>Pour la recherche des mots proches, on appliquera diverses heuristiques, basées sur la forme du mot :<br>repérage des bigrammes ou trigrammes impossibles en français<br>redoublement ou dé-doublement de consonnes<br>suppression/insertion de diacritiques<br>distance d'édition<br>etc<br>Ces heuristiques, qui peuvent être plus nombreuses, seront étudiées linguistiquement afin de déterminer précisément leurs conditions d'application. | Java | Marie Candito | | 2 personnes |
| | | Réaccentuation | Le programme est chargé de remettre les accents et autres signes diacritiques manquant dans un texte fourni en typographie dite pauvre.<br>On s'aidera d'un lexique de formes fléchies fourni. Bien sûr, certaines formes peuvent être réaccentuées directement, et d'autres sont ambigues. Une fois que l'algorithme principal sera établi, on envisagera des heuristiques pour lever les ambiguïtés. Exemple :<br>La ou le francais n'est pas accentue,<br>il y a de la gene,<br>mais quand le systeme m'accentue, | Java | Benoît Crabbé | | 2 personnes |

| ANNEE | NIVEAU | SUJET | DESCRIPTIF | LANGAGE | RESPONSABLE | DIFFICULTE | GROUPE |
|---|---|---|---|---|---|---|---|
| | | | je suis moins gene! | | | | |
| | | Extraction de sigles | L'objectif est d'être capable de construire, semi-automatiquement, la liste des sigles (acronymes) (par exemple EDF), et de leur forme développée (par exemple Electricité de France), utilisés dans un texte donné. Le programme aura deux fonctions :<br>Création (extraction) de la liste (triée, normalisée) des sigles du texte donné. Attention aux formes diverses d'un même sigle (Inalf, I.N.A.L.F, INaLF).<br>Proposition (interactive) de syntagmes candidats pour la forme développée, quand on en trouve dans le texte.<br>Le texte fourni est en ASCII brut | Java | Marie Candito | | 2 personnes |
| | | Justification et césure | Il s'agit de réaliser un programme qui justifie (au sens des traitements de texte) un texte fourni en ASCII, sur un nombre de colonnes donné, en ajoutant des espaces entre les mots et/ou en découpant les mots selon les règles en usage pour le français.<br>On supposera que tous les caractères ont la même dimension (« fonte fixe »).<br>Pour le découpage (éventuel) des mots (césure), on prendra bien garde de distinguer les règles, qui seront stockées dans un fichier, du programme lui-même. On s'autorisera à utiliser un dictionnaire d'exceptions. | Java | Benoît Crabbé | | 2 personnes |
| | M1 | Concordancier et outils statitistiques pour corpus annoté | On propose de réaliser une suite d'outils pour l'exploration de corpus annotés en morphosyntaxe. Celle-ci se composera de deux sous-composantes :<br>Une suite d'outils statistiques qui permettront d'obtenir des informations quantitatives sur le corpus, comme la fréquence et le nombre d'occurrences des mots. Un outil statistique qui proposera de détecter les collocations en corpus en utilisant une heuristique guidée par l'information mutuelle. Les outils statistiques devront produire une sortie texte qui permettra l'usage de leurs résultats dans un logiciel approprié au traitement de données comme Microsoft Excell ou R.<br>Un concordancier. Le concordancier permettra de faire des recherches sur le corpus annoté. Celui-ci sera réalisé en prenant en compte trois facteurs principaux (1) L'expressivité du langage de requête (2) L'efficacité de la recherche en terme de temps de réponse et (3) de fournir une interface utilisateur conviviale permettant notamment à celui-ci de trier les résultats d'une concordance. On veillera à ce que le concordancier soit relativement indépendant des formats de corpus sur lequel il sera développé. | Java | Benoît Crabbé | Très facile à très difficile | 2 personnes |
| | | Extraction de collocations en corpus | Les collocations sont des ensembles de deux ou plusieurs mots qui sont plus fréquemment coocurrents que la normale. Il s'agit de constructions qui ont une syntaxe classique mais une sémantique qui n'est pas complètement compositionnelle.<br>Par exemple on considère que thé fort est une collocation car ces deux mots apparaissent assez souvent en même temps et le sens de fort dans cette séquence est légèrement altéré : il ne s'agit pas d'une grande force physique mais plutôt d'une forte concentration d'un agent du thé.<br>Ce sujet propose de partir à la découverte de différentes méthodes connues de détection de collocations en corpus reposant sur des tests d'hypothèses statistiques (comme par exemple le test du Chi2 ou le test exact de Fisher) en utilisant des logiciels d'analyse de données appropriés. | Python ou perl et initiation à R | Benoît Crabbé | Moyen | 2 personnes |
| | | Repérage des entités nommées | On regroupe sous le terme "entités nommées" les noms de personnes, de lieux, de dates, noms d'entreprises, adresses, etc. Il s'agit d'expressions qui dénotent une entité unique de façon presque indépendante du contexte. On s'intéresse aux entités nommées pour plusieurs raisons :<br>elles constituent des syntagmes qui peuvent être relativement complexes au point de vue syntaxique (par exemple une adresse, ou un nom d'association) dont le repérage préalable peut grandement siplifier une analyse syntaxique ;<br>dans une perspective de recherche d'information, la reconnaissance des entités nommées permet de savoir de quoi parle un texte ;<br>elles sont nécessaires pour la résolution des anaphores. | Java | Marie Candito | Facile à très difficile | 2 ou 3 personnes |

| ANNEE | NIVEAU | SUJET | DESCRIPTIF | LANGAGE | RESPONSABLE | DIFFICULTE | GROUPE |
|---|---|---|---|---|---|---|---|
| | | | Il s'agit dans ce projet de repérer de la façon la plus complète possible dans un texte étiqueté ou non, les entités de type "personne". Pour cela, on envisagera un algorithme en deux étapes (qui peuvent se répéter) :<br><br>au moyen de règles générales et de dictionnaires spécialisés (noms propres, amorces --- c'est-à-dire mots qui introduisent systématiquement des entités nommées, comme 'Melle', etc.), constitution d'une "table des symboles" des entités présentes dans le texte ;<br><br>à partir de cette table des symboles, et en tenant compte des formes variées sous lesquelles une même entités peut être désignée, recherche de nouvelles entités, voire de nouvelles règles trouvées précédemment.<br><br>L'idée est que le programme s'enrichit au fur et à mesure qu'il est utilisé. | | | | |
| | | Réseaux sémantiques | À partir de définitions provenant de dictionnaires électroniques, préalablement étiquetées, il s'agit de construire un « réseau sémantique ».<br>Pour chaque entrée (qui peut correspondre à un des sens d'une lexie), on repèrera les mots pleins qui participent à sa définition (il faut donc (1) distinguer les mots pleins des mots « outils », (2) repérer la définition proprement dite parmi l'ensemble des informations associées à une entrée (catégorie, exemples, synonymes...)), et on construira un réseau reliant l'entrée à tous ces mots pleins.<br>Ce réseau pourra être stocké sous forme de fichier Ascii, dans un format du genre de celui de WordNet. On pourra aussi, éventuellement, proposer une interface graphique permettant de visualiser graphiquement le réseau, voire de le modifier.<br><br>Dans un deuxième temps, on réalisera un programme exploitant ce réseau pour désambiguïser les sens d'un mot en contexte. Principe : étant donnée une phrase contenant le mot concerné, on repère les mots pleins qui l'entourent (contexte), et on recherche ces mots dans le réseau. Alors le sens correspondant est celui qui est le plus proche (dans un sens qu'il faut précisément définir) des mots pleins de son contexte.<br><br>D'autres exploitations d'un tel réseau peuvent être envisagées. | Java | Pascal Amsili | Moyen à très difficile | 2 ou 3 personnes |
| | | La sous-catégorisation verbale en corpus | Il s'agit de compléter le French treebank du laboratoire LLF. voir http://www.llf.cnrs.fr/Gens/Abeille/French-Treebank-fr.php<br>A partir de Treelex, un lexique de valence extrait par A. Kupsc à partir de ce corpus (voir http://erssab.u-bordeaux3.fr/article.php3?id_article=150) il s'agit de marquer les attributs subcat pour chaque occurrence verbale dans le corpus en désambiguisant en contexte si plusieurs valences sont disponibles. Il s'agira d'un marquage automatique (langage de programmation libre, python recommandé) à valider manuellement. Il s'agira aussi de marquer la voix passive le cas échéant.<br>Le projet peut se concerner au marquage automatique, et la validation manuelle peut se faire dans le cadre d'un stage rémunéré par le laboratoire LLF (300 euros net/mois). Il est validé deux fois. | | Anne Abeillé, Marie Candito | Facile à moyen | 1 personne |
| | | Marquage des antécédents pronominaux en corpus | Il s'agit de tester des algorithmes de résolution d'anaphores sur texte analysé syntaxiquement (Le French treebank du laboratoire LLF pourra être utilisé, voir http://www.llf.cnrs.fr/Gens/Abeille/French-Treebank-fr.php<br><br>A partir des occurrences pronominales de 3e personne marquées pour le genre et le nombre de leurs antécédents, il s'agit de proposer un ou plusieurs candidats dans la même phrase (pour les relatifs et les réfléchis) ou non (pour les personnels).<br>Les mots antécédents seront repérés par leur numéro dans la phrase (ID) éventuellement préfixé par le numéro de la phrase. | Java ou python | Pascal Amsili | Facile à très difficile | 2 personnes |
| 2007-2008 | L3 | | | | | | |
| | M1 | | | | | | |
| 2006-2007 | L3 | | | | | | |
| | M1 | | | | | | |
| 2005-2006 | L3 | | | | | | |

| ANNEE | NIVEAU | SUJET | DESCRIPTIF | LANGAGE | RESPONSABLE | DIFFICULTE | GROUPE |
|---|---|---|---|---|---|---|---|
| 2006 | M1 | Repérage des entités nommées | On regroupe sous le terme "entités nommées" les noms de personnes, de lieux, de dates, noms d'entreprises, adresses, etc. Il s'agit d'expressions qui dénotent une entité unique de façon presque indépendante du contexte. On s'intéresse aux entités nommées pour plusieurs raisons :<br>o elles constituent des syntagmes qui peuvent être relativement complexes au point de vue syntaxique (par exemple une adresse, ou un nom d'association) dont le repérage préalable peut grandement siplifier une analyse syntaxique ;<br>o dans une perspective de recherche d'information, la reconnaissance des entités nommées permet de savoir de quoi parle un texte ;<br>o elles sont nécessaires pour la résolution des anaphores.<br>Il s'agit dans ce projet de repérer de la façon la plus complète possible dans un texte étiqueté ou non, les entités temporelles. Pour cela, on envisagera un algorithme en deux étapes (qui peuvent se répéter) :<br>o au moyen de règles générales et de dictionnaires spécialisés (noms propres, amorces --- c'est-à-dire mots qui introduisent ou suivent systématiquement des entités nommées, comme 'minutes', etc.), constitution d'une "table des symboles" des entités présentes dans le texte ;<br>o à partir de cette table des symboles, et en tenant compte des formes variées sous lesquelles une même entité peut être désignée, recherche de nouvelles entités, voire de nouvelles règles trouvées précédemment.<br>L'idée est que le programme s'enrichit au fur et à mesure qu'il est utilisé. | *Au choix parmi les langages étudiés en cours* | | | |
| | | Analyseur syntaxique d'une grammaire ambigüe (Earley) | Il s'agit d'implémenter, pour un fragment significatif du français (ou d'une autre langue), comportant des ambiguïtés, l'algorithme d'analyse d'Earley. | *Au choix parmi les langages étudiés en cours* | | | |
| | | Réseaux sémantiques | À partir de définitions provenant de dictionnaires électroniques, préalablement étiquetées, il s'agit de construire un « réseau sémantique ».<br>Pour chaque entrée (qui peut correspondre à un des sens d'une lexie), on repèrera les mots pleins qui participent à sa définition (il faut donc (1) distinguer les mots pleins des mots « outils », (2) repérer la définition proprement dite parmi l'ensemble des informations associées à une entrée (catégorie, exemples, synonymes...)), et on construira un réseau reliant l'entrée à tous ces mots pleins.<br>Ce réseau pourra être stocké sous forme de fichier Ascii, dans un format du genre de celui de WordNet.<br>On pourra aussi, éventuellement, proposer une interface graphique permettant de visualiser graphiquement le réseau, voire de le modifier.<br>Dans un deuxième temps, on réalisera un programme exploitant ce réseau pour désambiguïser les sens d'un mot en contexte. Principe : étant donnée une phrase contenant le mot concerné, on repère les mots pleins qui l'entourent (contexte), et on recherche ces mots dans le réseau. Alors le sens correspondant est celui qui est le plus proche (dans un sens qu'il faut précisément définir) des mots pleins de son contexte.<br>D'autres exploitations d'un tel réseau peuvent être envisagées. | *Au choix parmi les langages étudiés en cours* | | | |
| | | Détection de la langue d'un texte | Pour détecter la langue d'un texte, on peut constituer une base de connaissances à partir d'un corpus de textes classés par langue. Pour chaque langue, un premier programme (à écrire) recueillera des statistiques significatives, basées sur les lettres (par exemple, il y a plus de "w" en anglais qu'en français). Le choix du modèle probabiliste employé et de ses paramètres (bigrammes, trigrammes) devra être justifié dans le rapport. On peut en proposer plusieurs et discuter de leurs avantages et inconvenients (rapport entre précision de la reconnaissance et volume de la base de connaissances ou longueur du texte nécessaire pour reconnaître sa langue).<br>Un deuxième programme (à écrire), utilisera ces bases de connaissances pour reconnaître la langue d'un texte. | *Au choix parmi les langages étudiés en cours* | | | |

| ANNEE | NIVEAU | SUJET | DESCRIPTIF | LANGAGE | RESPONSABLE | DIFFICULTE | GROUPE |
|---|---|---|---|---|---|---|---|
| | | | Vous utiliserez un corpus d'apprentissage comprenant des textes plus ou moins variés d'un certain nombre de langues (le plus de langues possible, au moins 4). Le corpus de test ne devra pas contenir de texte appartenant au corpus d'apprentissage.<br>Amélioration possible : gérer des textes dans différents encodages, pour les langues à alphabet non latin. | | | | |
| | | Wordnet en java | | Java | | | |
| | | Résolution des anaphores personnelles | | *Au choix parmi les langages étudiés en cours* | | | |
| 2004-2005 | L3 | Correcteur orthographique (approche graphémique) | On se propose de réaliser un correcteur orthographique (lexical), qui, disposant d'un dictionnaire de formes fléchies, détecte les mots mal orthographiés, et propose si possible une correction.<br>Le programme prend un texte (ASCII brut) en entrée, et pour chaque forme non présente dans le dictionnaire, propose à l'utilisateur :<br>• de choisir un de ces remplaçants<br>o pour l'occurrence, ou pour toutes les occurrences<br>• de fournir un remplaçant<br>o pour l'occurrence, ou pour toutes les occurrences<br>o si le remplaçant n'est pas dans le lexique,<br>☐ de l'insérer dans le lexique pour ce texte,<br>☐ de l'insérer dans le lexique stable<br>• d'ignorer la correction, c'est-à-dire conserver le mot initial, pour cette occurrence ou pour toutes, en l'insérant ou non dans le lexique.<br>Les remplaçants proposés par le programme (premier cas plus haut) seront des mots figurant dans le dictionnaire, et ayant une certaine proximité avec le mot fautif.<br>Pour la recherche des mots proches, on appliquera diverses heuristiques, basées sur la forme du mot :<br>• repérage des bigrammes ou trigrammes impossibles en français<br>• redoublement ou dé-doublement de consonnes<br>• suppression/insertion de diacritiques<br>• distance d'édition<br>• etc.<br>Ces heuristiques, qui peuvent être plus nombreuses, seront étudiées linguistiquement afin de déterminer précisément leurs conditions d'application. | Pascal ou C | | | |
| | | Réaccentuation | Le programme est chargé de remettre les accents et autres signes diacritiques manquant dans un texte fourni en typographie dite pauvre.<br>On s'aidera d'un lexique de formes fléchies fourni. Bien sûr, certaines formes peuvent être réaccentuées directement, et d'autres sont ambigues. Une fois que l'algorithme principal sera établi, on envisagera des heuristiques pour lever les ambiguïtés. Exemple :<br>La ou le francais n'est pas accentue,<br>il y a de la gene,<br>mais quand le systeme m'accentue,<br>je suis moins gene! | Pascal ou C | | | |

| ANNEE | NIVEAU | SUJET | DESCRIPTIF | LANGAGE | RESPONSABLE | DIFFICULTE | GROUPE |
|---|---|---|---|---|---|---|---|
| | | Extraction de sigles | L'objectif est d'être capable de construire, semi-automatiquement, la liste des sigles (acronymes) (par exemple EDF), et de leur forme développée (par exemple Electricité de France), utilisés dans un texte donné. Le programme aura deux fonctions :<br>1. Création (extraction) de la liste (triée, normalisée) des sigles du texte donné. Attention aux formes diverses d'un même sigle (Inalf, I.N.A.L.F, INaLF).<br>2. Proposition (interactive) de syntagmes candidats pour la forme développée, quand on en trouve dans le texte.<br>Le texte fourni est en ASCII brut. | Pascal ou C | | | |
| | | Justification et césure | Il s'agit de réaliser un programme qui justifie (au sens des traitements de texte) un texte fourni en ASCII, sur un nombre de colonnes donné, en ajoutant des espaces entre les mots et/ou en découpant les mots selon les règles en usage pour le français.<br>On supposera que tous les caractères ont la même dimension (« fonte fixe »).<br>Pour le découpage (éventuel) des mots (césure), on prendra bien garde de distinguer les règles, qui seront stockées dans un fichier, du programme lui-même. On s'autorisera à utiliser un dictionnaire d'exceptions. | Pascal ou C | | | |
| | | Marquage des syntagmes nominaux | Il s'agit, dans un texte étiqueté, de repérer le début et la fin des syntagmes nominaux (ou d'une bonne partie d'entre eux), de manière à produire en sortie un texte dans lequel ces débuts et fin de SN seront balisés (par exemple <sn>le chat de <sn>la voisine</sn></sn>).<br>Le programme permettra à la fois une utilisation "batch", qui créera directement le fichier résultat, et une utilisation interactive, qui pemettra à l'utilisateur de voir les résultats produits au fur et à mesure.<br>Une étude linguistique va permettre de déterminer les séquences de lemmes ou d'étiquettes qui vont correspondre à un début ou une fin de syntagme. On fera en sorte que ces "règles" soient placées dans un fichier à part, et non "codées en dur", pour permettre une mise au point plus facile et une meilleure qualité du programme. | Pascal ou C | | | |
| | M1 | Repérage des entités nommées | On regroupe sous le terme "entités nommées" les noms de personnes, de lieux, de dates, noms d'entreprises, adresses, etc. Il s'agit d'expressions qui dénotent une entité unique de façon presque indépendante du contexte. On s'intéresse aux entités nommées pour plusieurs raisons :<br>• elles constituent des syntagmes qui peuvent être relativement complexes au point de vue syntaxique (par exemple une adresse, ou un nom d'association) dont le repérage préalable peut grandement siplifier une analyse syntaxique ;<br>• dans une perspective de recherche d'information, la reconnaissance des entités nommées permet de savoir de quoi parle un texte ;<br>• elles sont nécessaires pour la résolution des anaphores.<br>Il s'agit dans ce projet de repérer de la façon la plus complète possible dans un texte étiqueté ou non, les entités de type "personne". Pour cela, on envisagera un algorithme en deux étapes (qui peuvent se répéter) :<br>• au moyen de règles générales et de dictionnaires spécialisés (noms propres, amorces --- c'est-à-dire mots qui introduisent systématiquement des entités nommées, comme 'Melle', etc.), constitution d'une "table des symboles" des entités présentes dans le texte ;<br>• à partir de cette table des symboles, et en tenant compte des formes variées sous lesquelles une même entités peut être désignée, recherche de nouvelles entités, voire de nouvelles règles trouvées précédemment.<br>L'idée est que le programme s'enrichit au fur et à mesure qu'il est utilisé. | Lisp, Prolog ou java | | | |
| | | Analyseur syntaxique d'une grammaire ambigüe (Earley) | Il s'agit d'implémenter, pour un fragment significatif du français (ou d'une autre langue), comportant des ambiguïtés, l'algorithme d'analyse d'Earley. | Lisp, Prolog ou java | | | |

| ANNEE | NIVEAU | SUJET | DESCRIPTIF | LANGAGE | RESPONSABLE | DIFFICULTE | GROUPE |
|---|---|---|---|---|---|---|---|
| | | Réseaux sémantiques | À partir de définitions provenant de dictionnaires électroniques, préalablement étiquetées, il s'agit de construire un « réseau sémantique ».<br>Pour chaque entrée (qui peut correspondre à un des sens d'une lexie), on repèrera les mots pleins qui participent à sa définition (il faut donc (1) distinguer les mots pleins des mots « outils », (2) repérer la définition proprement dite parmi l'ensemble des informations associées à une entrée (catégorie, exemples, synonymes...)), et on construira un réseau reliant l'entrée à tous ces mots pleins.<br>Ce réseau pourra être stocké sous forme de fichier Ascii, dans un format du genre de celui de WordNet.<br>On pourra aussi, éventuellement, proposer une interface graphique permettant de visualiser graphiquement le réseau, voire de le modifier.<br>Dans un deuxième temps, on réalisera un programme exploitant ce réseau pour désambiguïser les sens d'un mot en contexte. Principe : étant donnée une phrase contenant le mot concerné, on repère les mots pleins qui l'entourent (contexte), et on recherche ces mots dans le réseau. Alors le sens correspondant est celui qui est le plus proche (dans un sens qu'il faut précisément définir) des mots pleins de son contexte.<br>D'autres exploitations d'un tel réseau peuvent être envisagées. | Lisp, Prolog ou java | | | |
| | | Détection de la langue d'un texte | Pour détecter la langue d'un texte, on peut constituer une base de connaissances à partir d'un corpus de textes classés par langue. Pour chaque langue, un premier programme (à écrire) recueillera des statistiques significatives, basées sur les lettres (par exemple, il y a plus de "w" en anglais qu'en français). Le choix du modèle probabiliste employé et de ses paramètres (bigrammes, trigrammes) devra être justifié dans le rapport. On peut en proposer plusieurs et discuter de leurs avantages et inconvenients (rapport entre précision de la reconnaissance et volume de la base de connaissances ou longueur du texte nécessaire pour reconnaître sa langue).<br>Un deuxième programme (à écrire), utilisera ces bases de connaissances pour reconnaître la langue d'un texte.<br>Vous utiliserez un corpus d'apprentissage comprenant des textes plus ou moins variés d'un certain nombre de langues (le plus de langues possible, au moins 4). Le corpus de test ne devra pas contenir de texte appartenant au corpus d'apprentissage.<br>Amélioration possible : gérer des textes dans différents encodages, pour les langues à alphabet non latin. | Lisp, Prolog ou java | | | |
| | | Wordnet en java | | Java | | | |
| 2003-2004 | L3 | | | | | | |
| | M1 | | | | | | |
| 2002-2003 | L3 | Correcteur orthographique (approche graphémique) | On se propose de réaliser un correcteur orthographique (lexical), qui, disposant d'un dictionnaire de formes fléchies, détecte les mots mal orthographiés, et propose si possible une correction.<br>Le programme prend un texte (ASCII brut) en entrée, et pour chaque forme non présente dans le dictionnaire, propose à l'utilisateur :<br>• de choisir un de ces remplaçants<br>o pour l'occurrence, ou pour toutes les occurrences<br>• de fournir un remplaçant<br>o pour l'occurrence, ou pour toutes les occurrences<br>o si le remplaçant n'est pas dans le lexique,<br>☐ de l'insérer dans le lexique pour ce texte,<br>☐ de l'insérer dans le lexique stable<br>• d'ignorer la correction, c'est-à-dire conserver le mot initial, pour cette occurrence ou pour toutes, en l'insérant ou non dans le lexique. | Pascal ou C | | | 2 ou 3 personnes |

| ANNEE | NIVEAU | SUJET | DESCRIPTIF | LANGAGE | RESPONSABLE | DIFFICULTE | GROUPE |
|---|---|---|---|---|---|---|---|
| | | | Les remplaçants proposés par le programme (premier cas plus haut) seront des mots figurant dans le dictionnaire, et ayant une certaine proximité avec le mot fautif.<br>Pour la recherche des mots proches, on appliquera diverses heuristiques, basées sur la forme du mot :<br>• repérage des bigrammes ou trigrammes impossibles en français<br>• redoublement ou dé-doublement de consonnes<br>• suppression/insertion de diacritiques<br>• distance d'édition<br>• etc.<br>Ces heuristiques, qui peuvent être plus nombreuses, seront étudiées linguistiquement afin de déterminer précisément leurs conditions d'application. | | | | |
| | | Réaccentuation | Le programme est chargé de remettre les accents et autres signes diacritiques manquant dans un texte fourni en typographie dite pauvre.<br>On s'aidera d'un lexique de formes fléchies fourni. Bien sûr, certaines formes peuvent être réaccentuées directement, et d'autres sont ambigues. Une fois que l'algorithme principal sera établi, on envisagera des heuristiques pour lever les ambiguïtés. Exemple :<br>La ou le francais n'est pas accentue,<br>il y a de la gene,<br>mais quand le systeme m'accentue,<br>je suis moins gene!<br>On pourra s'inspirer du programme « Reacc » développé par le RALI à l'Université de Montréal. | Pascal ou C | | | 2 ou 3 personnes |
| | | Recherche de patrons | L'objectif de ce projet est de réaliser un programme permettant de détecter dans un corpus étiqueté et lemmatisé des séquences de mots conformes à des patrons lexico-syntaxiques. Un patron étant une suite de mots, de lemmes ou de catégories morpho-syntaxiques. Exemples: « N à N », « relation de N »<br>Le programme prendra en entrée un corpus et un fichier de patrons et produira en sortie une liste classée par fréquence, de toutes les séquences de mots conformes aux patrons, accompagnées de leur nombre d'occurrences.<br>Dans la définition des patrons, il faudra introduire le moyen de distinguer les mots des lemmes et des catégories.<br>On attachera un soin particulier au choix de la structure de données, et à l'algorithme, qui doit faire le calcul en un temps raisonnable. | Pascal ou C | | | 2 ou 3 personnes |
| | | Extraction de sigles | L'objectif est d'être capable de construire, semi-automatiquement, la liste des sigles (acronymes) (par exemple EDF), et de leur forme développée (par exemple Electricité de France), utilisés dans un texte donné. Le programme aura deux fonctions :<br>1. Création (extraction) de la liste (triée, normalisée) des sigles du texte donné. Attention aux formes diverses d'un même sigle (Inalf, I.N.A.L.F, INaLF).<br>2. Proposition (interactive) de syntagmes candidats pour la forme développée, quand on en trouve dans le texte.<br>Le texte fourni est en ASCII brut. | Pascal ou C | | | 2 ou 3 personnes |
| | | Justification et césure | Il s'agit de réaliser un programme qui justifie (au sens des traitements de texte) un texte fourni en ASCII, sur un nombre de colonnes donné, en ajoutant des espaces entre les mots et/ou en découpant les mots selon les règles en usage pour le français.<br>On supposera que tous les caractères ont la même dimension (« fonte fixe »).<br>Pour le découpage (éventuel) des mots (césure), on prendra bien garde de distinguer les règles, qui seront stockées dans un fichier, du programme lui-même. On s'autorisera à utiliser un dictionnaire d'exceptions. | Pascal ou C | | | 2 ou 3 personnes |

| ANNEE | NIVEAU | SUJET | DESCRIPTIF | LANGAGE | RESPONSABLE | DIFFICULTE | GROUPE |
|---|---|---|---|---|---|---|---|
| | | Enrichissement de requêtes | Il s'agit de se placer dans la problématique suivante : on formule des requêtes à un moteur de recherche, les requêtes pouvant contenir un ou plusieurs mots, voire un syntagme ou une phrase. Sachant que les taux de réussite de telles requêtes ne sont pas toujours satisfaisants, on se propose d'enrichir ces requêtes (en ajoutant des mots, ou en utilisant les opérateurs classiques de moteurs de recherche...) pour améliorer le résultat.<br>Cet enrichissement sera réalisé sur la bases d'information de nature lexicale ou terminologique, stockées dans un réseau sémantique approprié.<br>Le but du projet n'est pas de réaliser un tel réseau sémantique à grande couverture, mais de spécifier, pour un domaine conceptuel fixé, les heuristiques à utiliser et l'architecture du réseau correspondant. Le projet donnera lieu à la réalisation d'une maquette, au sens industriel du terme. | Pascal ou C | | | 2 ou 3 personnes |
| | M1 | Repérage des "entités nommées" | On regroupe sous le terme « entités nommées » les noms de personnes, de lieux, les dates, noms d'entreprises, adresses, etc. Le but du projet consiste à repérer automatiquement un sous-ensemble homogène parmi les entités (par exemple les noms de personnes ou les noms d'entreprises), l'idée étant de faciliter un traitement ultérieur, sémantique (extraction d'information, résolution d'anaphore...), syntaxique (analyse, chunking...), etc.<br>Le programme envisagé procède par étapes successives sur le même texte. Dans une première étape, grâce aux dictionnaires spécialisés (cf. plus loin), on étiquette les mots simples. On partira d'un corpus déjà étiqueté morpho-syntaxiquement. La seconde étape, basée sur des règles, permettra de regrouper ces mots simples pour former des entités nommées. Enfin, une troisième étape, facultative dans ce projet, pourrait tenter de relier entre elles les entités co-référentes. Les éléments seront marqués dans le texte au moyen d'un système de balisage de type SGML.<br>Le système repose donc, outre le programme lui-même, sur les ressources suivantes :<br>• des dictionnaires :<br>o dictionnaire de la langue générale pour repérer les mots inconnus.<br>o dictionnaires spécialisés de prénoms, de noms de sociétés connues, etc.<br>o dictionnaires spécialisés d'« amorces », mots ou groupes de mots qui peuvent marquer le début d'une entité nommée.<br>• une « grammaire » permettant de combiner des mots repérés de natures diverses (mots inconnus, prénoms, amorces...) pour en faire une entité (un nom de personne).<br>Les dictionnaires et la grammaire sont stockés dans des fichiers textes. On assure ainsi une séparation nette entre les données et les traitement (le programme).<br>Exemple<br>Soit le texte « M. Jack Lang a gagné la mairie. »<br>"M." est stocké dans un dictionnaire "d'amorces"<br>"Jack" est stocké dans un dictionnaire de prénoms<br>"Lang" est inconnu du système<br>La première étape permet d'obtenir :<br><TR_PERSON>M.</TR_PERSON> <PERSON>Jack</PERSON> <UFIRSTUNKNOWN>Lang</UFIRSTUNKNOWN> a gagné la mairie.<br>Une règle de la forme<br>TR_PERSON PERSON UFIRSTUNKNOWN ==> PERSON<br>va pouvoir être activée.<br>D'où le résultat de la seconde étape :<br><PERSON>M. Jack Lang</PERSON> a gagné la mairie.<br>Une troisième étape permettrait d'identifier toutes les occurrences de Jack Lang | Lisp, Prolog ou java | | | 2 ou 3 personnes |

| ANNEE | NIVEAU | SUJET | DESCRIPTIF | LANGAGE | RESPONSABLE | DIFFICULTE | GROUPE |
|---|---|---|---|---|---|---|---|
| | | | <PERSON id=1>M. Jack Lang</PERSON> a gagné la mairie. | | | | |
| | | Réseaux sémantiques | À partir de définitions provenant de dictionnaires électroniques, préalablement étiquetées, il s'agit de construire un « réseau sémantique ».<br>Pour chaque entrée (qui peut correspondre à un des sens d'une lexie), on repèrera les mots pleins qui participent à sa définition (il faut donc (1) distinguer les mots pleins des mots « outils », (2) repérer la définition proprement dite parmi l'ensemble des informations associées à une entrée (catégorie, exemples, synonymes...)), et on construira un réseau reliant l'entrée à tous ces mots pleins.<br>Ce réseau pourra être stocké sous forme de fichier Ascii, dans un format du genre de celui de WordNet.<br>On pourra aussi, éventuellement, proposer une interface graphique permettant de visualiser graphiquement le réseau, voire de le modifier.<br>Dans un deuxième temps, on réalisera un programme exploitant ce réseau pour désambiguïser les sens d'un mot en contexte. Principe : étant donnée une phrase contenant le mot concerné, on repère les mots pleins qui l'entourent (contexte), et on recherche ces mots dans le réseau. Alors le sens correspondant est celui qui est le plus proche (dans un sens qu'il faut précisément définir) des mots pleins de son contexte.<br>D'autres exploitations d'un tel réseau peuvent être envisagées. | Lisp, Prolog ou java | | | 2 ou 3 personnes |
| | | "Complétion" automatique | On se propose de réaliser un programme qui accélère la vitesse de frappe d'un utilisateur en proposant - de façon intelligente - la fin des mots en cours de frappe.<br>Un tel système, pour être vraiment performant, devrait faire de la prédiction syntaxique en temps réel. On se contentera dans ce projet de faire un système qui propose la suite la plus probable des mots.<br>La méthode repose sur un calcul de probabilité basé sur la fréquence d'apparition de chaque mot dans un corpus préalablement fourni. On construit un arbre dont chaque chemin correspond à un mot (une lettre par arc), et dont chaque arc porte une probabilité.<br>Par exemple, la probabilité de la prochaine lettre de la suite ``v-o-l-u'' est respectivement de 0/9, 8/9 (3+5/9), 1/9, 0/9 pour les lettres `b', `m', `p' et `t' si nous avons dans le corpus :<br>0 occurrences du mot volubile<br>3 " volume<br>5 " volumineux<br>1 " voluptueux<br>0 " volute<br>La lettre la plus probable, ici ``m'', sera proposée à l'utilisateur, et un nouveau calcul est effectué pour la suite de lettres ``v-o-l-u-m''.<br>Un soin particulier sera apporté à l'interface de ce programme : l'utilisateur doit pouvoir accepter ou revenir facilement sur les propositions du système, etc. La dimension ergonomique de ce projet est au moins aussi importante que la dimension algorithmique : en particulier, il conviendra de choisir entre la stratégie qui consiste à ne produire automatiquement qu'une lettre à la fois, auquel cas notre objectif d'économie en vitesse de frappe est compromis et celle qui consiste à produire le suffixe en entier (à partir de quelle longueur de préfixe ?), au risque de nécessiter beaucoup de corrections de la part de l'utilisateur.<br>Un langage particulièrement adapté au projet serait Emacs Lisp, ce qui permettrait d'intégrer la complétion à l'éditeur Emacs, ce qui n'exclut cependant pas la réalisation de ce projet dans un autre langage. | Emacs Lisp | | | 2 ou 3 personnes |
| | | Étiqueteur morpho-syntaxique basé sur un tri-gramme | L'étiqueteur morpho-syntaxique développé dans le cadre du cours LI031 repose sur un modèle bi-gramme : la prédiction de la catégorie d'un mot dépend des catégories possibles du mot et de la catégorie du mot précédent. L'objectif de ce projet est de modifier l'étiqueteur pour que la prédiction repose sur la catégorie des deux mots précédents. | Lisp, Prolog ou java | | | 2 ou 3 personnes |

| ANNEE | NIVEAU | SUJET | DESCRIPTIF | LANGAGE | RESPONSABLE | DIFFICULTE | GROUPE |
|---|---|---|---|---|---|---|---|
| | | | Les modifications porteront d'une part sur les structures de données permettant de stocker les tri-grammes de façon efficace et d'autre part sur l'implémentation des méthodes de repli et d'interpolation permettant de pallier l'insuffisance des données d'apprentissage, plus importantes que dans le cas d'un bi-gramme. | | | | |
| | | Traitement des mots inconnus dans un étiqueteur morpho-syntaxique | L'étiqueteur morpho-syntaxique développé dans le cadre du cours LI031 ne prévoit pas le cas où un mot d'une phrase à étiqueter ne figure pas dans le lexique de l'étiqueteur. L'objectif de ce projet est d'intégrer à l'étiqueteur un module de gestion des mots inconnus.<br>Plusieurs solutions seront testées, parmi lesquelles, la prédiction de la catégorie du mot inconnu en fonction de sa forme (en particulier, les préfixes et les suffixes) et en fonction de son contexte (les catégories possibles des mots suivants et précédents) ou encore de la loi de probabilité qu'un mot inconnu appartienne aux différentes catégories (un mot inconnu à une plus grande probabilité d'être un nom prore qu'un article.). Les différentes solutions seront testées isolément et conjointement, une analyse des erreurs permettra de déterminer les points forts et les points faibles des différentes solutions. | Lisp, Prolog ou java | | | 2 ou 3 personnes |
| | | Désambiguïsation par automates | Partant d'un texte dont chaque mot est étiqueté avec toutes ses catégories morpho-syntaxiques possibles (étiquetage sans désambiguïsation), on va s'attacher à lever certaines ambiguïtés non pertinentes, c'est-à-dire à supprimer certaines des étiquettes attachées à certains mots.<br>L'idée est de disposer d'un programme capable de prendre en entrée des règles comme la suivante : « si X est marqué N et V, et que X est précédé d'un Y marqué Det, alors l'étiquette V doit être ôtée à X » ; et d'appliquer ces règles sur la totalité du corpus.<br>On conseille d'implémenter ces règles sous forme d'automates.<br>Le travail comprend donc une partie linguistique, visant à mettre en évidence des règles de désambiguïsation pour le français, basées sur les étiquettes et/ou les lemmes des mots précédents et suivants les mots ambigus une partie formelle, qui s'attachera à établir un format de règles correspondant aux besoins relevés par l'étude linguistique et une partir informatique, qui permettra à partir d'un corpus quelconque, et d'un fichier de règles, de désambiguïser le corpus (ou du moins d'en diminuer l'ambiguïté). | Lisp, Prolog ou java | | | 2 ou 3 personnes |
| 2001-2002 | L3 | Correcteur orthographique (approche graphémique) | On se propose de réaliser un correcteur orthographique (lexical), qui, disposant d'un dictionnaire de formes fléchies, détecte les mots mal orthographiés, et propose si possible une correction.<br>Le programme prend un texte (ASCII brut) en entrée, et pour chaque forme non présente dans le dictionnaire, propose à l'utilisateur :<br>• de choisir un de ces remplaçants<br>o pour l'occurrence, ou pour toutes les occurrences<br>• de fournir un remplaçant<br>o pour l'occurrence, ou pour toutes les occurrences<br>o si le remplaçant n'est pas dans le lexique,<br>☐ de l'insérer dans le lexique pour ce texte,<br>☐ de l'insérer dans le lexique stable<br>• d'ignorer la correction, c'est-à-dire conserver le mot initial, pour cette occurrence ou pour toutes, en l'insérant ou non dans le lexique.<br>Les remplaçants proposés par le programme (premier cas plus haut) seront des mots figurant dans le dictionnaire, et ayant une certaine proximité avec le mot fautif.<br>Pour la recherche des mots proches, on appliquera diverses heuristiques, basées sur la forme du mot :<br>• repérage des bigrammes ou trigrammes impossibles en français<br>• redoublement ou dé-doublement de consonnes<br>• suppression/insertion de diacritiques<br>• distance d'édition | Pascal ou C | | | |

| ANNEE | NIVEAU | SUJET | DESCRIPTIF | LANGAGE | RESPONSABLE | DIFFICULTE | GROUPE |
|---|---|---|---|---|---|---|---|
| | | | • etc.<br>Ces heuristiques, qui peuvent être plus nombreuses, seront étudiées linguistiquement afin de déterminer précisément leurs conditions d'application. | | | | |
| | | Réaccentuation | Le programme est chargé de remettre les accents et autres signes diacritiques manquant dans un texte fourni en typographie dite pauvre.<br>On s'aidera d'un lexique de formes fléchies fourni. Bien sûr, certaines formes peuvent être réaccentuées directement, et d'autres sont ambigues. Une fois que l'algorithme principal sera établi, on envisagera des heuristiques pour lever les ambiguïtés. Exemple :<br>La ou le francais n'est pas accentue,<br>il y a de la gene,<br>mais quand le systeme m'accentue,<br>je suis moins gene!<br>On pourra s'inspirer du programme « Reacc » développé par le RALI à l'Université de Montréal. | Pascal ou C | | | |
| | | Recherche de patrons | L'objectif de ce projet est de réaliser un programme permettant de détecter dans un corpus étiqueté et lemmatisé des séquences de mots conformes à des patrons lexico-syntaxiques. Un patron étant une suite de mots, de lemmes ou de catégories morpho-syntaxiques. Exemples: « N à N », « relation de N »<br>Le programme prendra en entrée un corpus et un fichier de patrons et produira en sortie une liste classée par fréquence, de toutes les séquences de mots conformes aux patrons, accompagnées de leur nombre d'occurrences.<br>Dans la définition des patrons, il faudra introduire le moyen de distinguer les mots des lemmes et des catégories.<br>On attachera un soin particulier au choix de la structure de données, et à l'algorithme, qui doit faire le calcul en un temps raisonnable. | Pascal ou C | | | |
| | | Extraction de sigles | L'objectif est d'être capable de construire, semi-automatiquement, la liste des sigles (acronymes) (par exemple EDF), et de leur forme développée (par exemple Electricité de France), utilisés dans un texte donné. Le programme aura deux fonctions :<br>1. Création (extraction) de la liste (triée, normalisée) des sigles du texte donné. Attention aux formes diverses d'un même sigle (Inalf, I.N.A.L.F, INaLF).<br>2. Proposition (interactive) de syntagmes candidats pour la forme développée, quand on en trouve dans le texte.<br>Le texte fourni est en ASCII brut. | Pascal ou C | | | |
| | | Justification et césure | Il s'agit de réaliser un programme qui justifie (au sens des traitements de texte) un texte fourni en ASCII, sur un nombre de colonnes donné, en ajoutant des espaces entre les mots et/ou en découpant les mots selon les règles en usage pour le français.<br>On supposera que tous les caractères ont la même dimension (« fonte fixe »).<br>Pour le découpage (éventuel) des mots (césure), on prendra bien garde de distinguer les règles, qui seront stockées dans un fichier, du programme lui-même. On s'autorisera à utiliser un dictionnaire d'exceptions. | Pascal ou C | | | |
| | | Enrichissement de requêtes | Il s'agit de se placer dans la problématique suivante : on formule des requêtes à un moteur de recherche, les requêtes pouvant contenir un ou plusieurs mots, voire un syntagme ou une phrase. Sachant que les taux de réussite de telles requêtes ne sont pas toujours satisfaisants, on se propose d'enrichir ces requêtes (en ajoutant des mots, ou en utilisant les opérateurs classiques de moteurs de recherche...) pour améliorer le résultat.<br>Cet enrichissement sera réalisé sur la bases d'information de nature lexicale ou terminologique, stockées dans un réseau sémantique approprié. | Pascal ou C | | | |

| ANNEE | NIVEAU | SUJET | DESCRIPTIF | LANGAGE | RESPONSABLE | DIFFICULTE | GROUPE |
|---|---|---|---|---|---|---|---|
| | | | Le but du projet n'est pas de réaliser un tel réseau sémantique à grande couverture, mais de spécifier, pour un domaine conceptuel fixé, les heuristiques à utiliser et l'architecture du réseau correspondant. Le projet donnera lieu à la réalisation d'une maquette, au sens industriel du terme. | | | | |
| | | Téléphone | La plupart des téléphones associent à chaque touche un certain nombre de lettres, permettant ainsi de transmettre des messages. Du fait qu'un chiffre ne correspond pas à une seule lettre, une suite de chiffres peut être ambiguë (elle peut correspondre à plus d'un mot). La suite 7-6-8-7, par exemple, correspond aux mots pour, sous et pots.<br>Le but du projet est de réaliser un programme qui prend en entrée une suite de chiffre et un dictionnaire de formes fléchies et qui propose en sortie la liste des mots du dictionnaire qui correspondent à la suite des chiffres. Ces mots seront classés selon une fréquence représentée dans le dictionnaire.<br>On attachera un soin particulier au choix de la structure de données, et à l'algorithme, qui doit faire le calcul en un temps raisonnable. | Pascal ou C | | | |
| | M1 | Repérage des "entités nommées" | On regroupe sous le terme « entités nommées » les noms de personnes, de lieux, les dates, noms d'entreprises, adresses, etc. Le but du projet consiste à repérer automatiquement un sous-ensemble homogène parmi les entités (par exemple les noms de personnes ou les noms d'entreprises), l'idée étant de faciliter un traitement ultérieur, sémantique (extraction d'information, résolution d'anaphore...), syntaxique (analyse, chunking...), etc.<br>Le programme envisagé procède par étapes successives sur le même texte. Dans une première étape, grâce aux dictionnaires spécialisés (cf. plus loin), on étiquette les mots simples. On partira d'un corpus déjà étiqueté morpho-syntaxiquement. La seconde étape, basée sur des règles, permettra de regrouper ces mots simples pour former des entités nommées. Enfin, une troisième étape, facultative dans ce projet, pourrait tenter de relier entre elles les entités co-référentes. Les éléments seront marqués dans le texte au moyen d'un système de balisage de type SGML.<br>Le système repose donc, outre le programme lui-même, sur les ressources suivantes :<br>• des dictionnaires :<br>o dictionnaire de la langue générale pour repérer les mots inconnus.<br>o dictionnaires spécialisés de prénoms, de noms de sociétés connues, etc.<br>o dictionnaires spécialisés d'« amorces », mots ou groupes de mots qui peuvent marquer le début d'une entité nommée.<br>• une « grammaire » permettant de combiner des mots repérés de natures diverses (mots inconnus, prénoms, amorces...) pour en faire une entité (un nom de personne).<br>Les dictionnaires et la grammaire sont stockés dans des fichiers textes. On assure ainsi une séparation nette entre les données et les traitement (le programme).<br>Exemple<br>Soit le texte « M. Jack Lang a gagné la mairie. »<br>"M." est stocké dans un dictionnaire "d'amorces"<br>"Jack" est stocké dans un dictionnaire de prénoms<br>"Lang" est inconnu du système<br>La première étape permet d'obtenir :<br><TR_PERSON>M.</TR_PERSON> <PERSON>Jack</PERSON> <UFIRSTUNKNOWN>Lang</UFIRSTUNKNOWN> a gagné la mairie.<br>Une règle de la forme<br>TR_PERSON PERSON UFIRSTUNKNOWN ==> PERSON<br>va pouvoir être activée.<br>D'où le résultat de la seconde étape : | Lisp, Prolog, Pascal, C, C++ ou java | | | |

| ANNEE | NIVEAU | SUJET | DESCRIPTIF | LANGAGE | RESPONSABLE | DIFFICULTE | GROUPE |
|---|---|---|---|---|---|---|---|
| | | | <PERSON>M. Jack Lang</PERSON> a gagné la mairie.<br>Une troisième étape permettrait d'identifier toutes les occurrences de Jack Lang<br><PERSON id=1>M. Jack Lang</PERSON> a gagné la mairie. | | | | |
| | | Réseaux sémantiques | À partir de définitions provenant de dictionnaires électroniques, préalablement étiquetées, il s'agit de construire un « réseau sémantique ».<br>Pour chaque entrée (qui peut correspondre à un des sens d'une lexie), on repèrera les mots pleins qui participent à sa définition (il faut donc (1) distinguer les mots pleins des mots « outils », (2) repérer la définition proprement dite parmi l'ensemble des informations associées à une entrée (catégorie, exemples, synonymes...)), et on construira un réseau reliant l'entrée à tous ces mots pleins.<br>Ce réseau pourra être stocké sous forme de fichier Ascii, dans un format du genre de celui de WordNet.<br>On pourra aussi, éventuellement, proposer une interface graphique permettant de visualiser graphiquement le réseau, voire de le modifier.<br>Dans un deuxième temps, on réalisera un programme exploitant ce réseau pour désambiguïser les sens d'un mot en contexte. Principe : étant donnée une phrase contenant le mot concerné, on repère les mots pleins qui l'entourent (contexte), et on recherche ces mots dans le réseau. Alors le sens correspondant est celui qui est le plus proche (dans un sens qu'il faut précisément définir) des mots pleins de son contexte.<br>D'autres exploitations d'un tel réseau peuvent être envisagées. | Lisp, Prolog, Pascal, C, C++ ou java | | | |
| | | "Complétion" automatique | On se propose de réaliser un programme qui accélère la vitesse de frappe d'un utilisateur en proposant - de façon intelligente - la fin des mots en cours de frappe.<br>Un tel système, pour être vraiment performant, devrait faire de la prédiction syntaxique en temps réel. On se contentera dans ce projet de faire un système qui propose la suite la plus probable des mots.<br>La méthode repose sur un calcul de probabilité basé sur la fréquence d'apparition de chaque mot dans un corpus préalablement fourni. On construit un arbre dont chaque chemin correspond à un mot (une lettre par arc), et dont chaque arc porte une probabilité.<br>Par exemple, la probabilité de la prochaine lettre de la suite ``v-o-l-u'' est respectivement de 0/9, 8/9 (3+5/9), 1/9, 0/9 pour les lettres `b', `m', `p' et `t' si nous avons dans le corpus :<br>0 occurrences du mot volubile<br>3 " volume<br>5 " volumineux<br>1 " voluptueux<br>0 " volute<br>La lettre la plus probable, ici ``m'', sera proposée à l'utilisateur, et un nouveau calcul est effectué pour la suite de lettres ``v-o-l-u-m''.<br>Un soin particulier sera apporté à l'interface de ce programme : l'utilisateur doit pouvoir accepter ou revenir facilement sur les propositions du système, etc. La dimension ergonomique de ce projet est au moins aussi importante que la dimension algorithmique : en particulier, il conviendra de choisir entre la stratégie qui consiste à ne produire automatiquement qu'une lettre à la fois, auquel cas notre objectif d'économie en vitesse de frappe est compromis et celle qui consiste à produire le suffixe en entier (à partir de quelle longueur de préfixe ?), au risque de nécessiter beaucoup de corrections de la part de l'utilisateur.<br>Un langage particulièrement adapté au projet serait Emacs Lisp, ce qui permettrait d'intégrer la complétion à l'éditeur Emacs, ce qui n'exclut cependant pas la réalisation de ce projet dans un autre langage. | Emacs Lisp | | | |
| | | Étiqueteur morpho-syntaxique basé sur un tri-gramme | L'étiqueteur morpho-syntaxique développé dans le cadre du cours LI031 repose sur un modèle bi-gramme : la prédiction de la catégorie d'un mot dépend des catégories possibles du mot et de la catégorie du mot précédent. L'objectif de ce projet est de modifier l'étiqueteur pour que la prédiction repose sur la catégorie des deux mots précédents. | Lisp, Prolog, Pascal, C, C++ ou java | | | |

| ANNEE | NIVEAU | SUJET | DESCRIPTIF | LANGAGE | RESPONSABLE | DIFFICULTE | GROUPE |
|---|---|---|---|---|---|---|---|
| | | | Les modifications porteront d'une part sur les structures de données permettant de stocker les tri-grammes de façon efficace et d'autre part sur l'implémentation des méthodes de repli et d'interpolation permettant de pallier l'insuffisance des données d'apprentissage, plus importantes que dans le cas d'un bi-gramme. | | | | |
| | | Traitement des mots inconnus dans un étiqueteur morpho-syntaxique | L'étiqueteur morpho-syntaxique développé dans le cadre du cours LI031 ne prévoit pas le cas où un mot d'une phrase à étiqueter ne figure pas dans le lexique de l'étiqueteur. L'objectif de ce projet est d'intégrer à l'étiqueteur un module de gestion des mots inconnus.<br>Plusieurs solutions seront testées, parmi lesquelles, la prédiction de la catégorie du mot inconnu en fonction de sa forme (en particulier, les préfixes et les suffixes) et en fonction de son contexte (les catégories possibles des mots suivants et précédents) ou encore de la loi de probabilité qu'un mot inconnu appartienne aux différentes catégories (un mot inconnu à une plus grande probabilité d'être un nom prore qu'un article.). Les différentes solutions seront testées isolément et conjointement, une analyse des erreurs permettra de déterminer les points forts et les points faibles des différentes solutions. | Lisp, Prolog, Pascal, C, C++ ou java | | | |
| | | Désambiguïsation par automates | Partant d'un texte dont chaque mot est étiqueté avec toutes ses catégories morpho-syntaxiques possibles (étiquetage sans désambiguïsation), on va s'attacher à lever certaines ambiguïtés non pertinentes, c'est-à-dire à supprimer certaines des étiquettes attachées à certains mots.<br>L'idée est de disposer d'un programme capable de prendre en entrée des règles comme la suivante : « si X est marqué N et V, et que X est précédé d'un Y marqué Det, alors l'étiquette V doit être ôtée à X » ; et d'appliquer ces règles sur la totalité du corpus.<br>On conseille d'implémenter ces règles sous forme d'automates.<br>Le travail comprend donc une partie linguistique, visant à mettre en évidence des règles de désambiguïsation pour le français, basées sur les étiquettes et/ou les lemmes des mots précédents et suivants les mots ambigus une partie formelle, qui s'attachera à établir un format de règles correspondant aux besoins relevés par l'étude linguistique et une partir informatique, qui permettra à partir d'un corpus quelconque, et d'un fichier de règles, de désambiguïser le corpus (ou du moins d'en diminuer l'ambiguïté). | Lisp, Prolog, Pascal, C, C++ ou java | | | |
| 2000-2001 | L3 | Correcteur orthographique (approche graphémique) | On se propose de réaliser un correcteur orthographique (lexical), qui, disposant d'un dictionnaire de formes fléchies, détecte les mots mal orthographiés, et propose si possible une correction.<br>Le programme prend un texte (ASCII brut) en entrée, et pour chaque forme non présente dans le dictionnaire, propose à l'utilisateur :<br>• de choisir un de ces remplaçants<br>o pour l'occurrence, ou pour toutes les occurrences<br>• de fournir un remplaçant<br>o pour l'occurrence, ou pour toutes les occurrences<br>o si le remplaçant n'est pas dans le lexique,<br>♣ de l'insérer dans le lexique pour ce texte,<br>♣ de l'insérer dans le lexique stable<br>• d'ignorer la correction, c'est-à-dire conserver le mot initial, pour cette occurrence ou pour toutes, en l'insérant ou non dans le lexique.<br>Les remplaçants proposés par le programme (premier cas plus haut) seront des mots figurant dans le dictionnaire, et ayant une certaine proximité avec le mot fautif.<br>Pour la recherche des mots proches, on appliquera diverses heuristiques, basées sur la forme du mot :<br>• repérage des bigrammes ou trigrammes impossibles en français<br>• redoublement ou dé-doublement de consonnes<br>• suppression/insertion de diacritiques<br>• insertion de lettres | Pascal ou C | | | |

| ANNEE | NIVEAU | SUJET | DESCRIPTIF | LANGAGE | RESPONSABLE | DIFFICULTE | GROUPE |
|---|---|---|---|---|---|---|---|
| | | | • interversion de lettres<br>• etc.<br>Ces heuristiques, qui peuvent être plus nombreuses, seront étudiées linguistiquement afin de déterminer précisément leurs conditions d'application. | | | | |
| | | Réaccentuation | Le programme est chargé de remettre les accents et autres signes diacritiques manquant dans un texte fourni en typographie dite pauvre.<br>On s'aidera d'un lexique de formes fléchies fourni. Bien sûr, certaines formes peuvent être réaccentuées directement, et d'autres sont ambigues. Une fois que l'algorithme principal sera établi, on envisagera des heuristiques pour lever les ambiguïtés. Exemple :<br>La ou le francais n'est pas accentue,<br>il y a de la gene,<br>mais quand le systeme m'accentue,<br>je suis moins gene!<br>On pourra s'inspirer du programme « Reacc » développé par le RALI à l'Université de Montréal. | Pascal ou C | | | |
| | | Calcul de collocations | On veut réaliser un programme de calcul des collocations dans un corpus étiqueté et lemmatisé.<br>Le but est de repérer et de compter les blocs de 2 ou 3 mots juxtaposés qui se retrouvent au moins deux fois dans le corpus. On s'intéressera aussi aux groupes « N à N », « N de N », etc., dont on calculera aussi les fréquences.<br>La sortie du programme consiste donc en une liste, classée par fréquence, de toutes les collocations qui y apparaissent au moins deux fois.<br>On attachera un soin particulier au choix de la structure de données, et à l'algorithme, qui doit faire le calcul en un temps raisonnable. | Pascal ou C | | | |
| | | Extraction de sigles | L'objectif est d'être capable de construire, semi-automatiquement, la liste des sigles (acronymes) (par exemple EDF), et de leur forme développée (par exemple Electricité de France), utilisés dans un texte donné. Le programme aura deux fonctions :<br>1. Création (extraction) de la liste (triée, normalisée) des sigles du texte donné. Attention aux formes diverses d'un même sigle (Inalf, I.N.A.L.F, INaLF).<br>2. Proposition (interactive) de syntagmes candidats pour la forme développée, quand on en trouve dans le texte.<br>Le texte fourni est en ASCII brut. | Pascal ou C | | | |
| | | Justification et "hyphénation" | Il s'agit de réaliser un programme qui justifie (au sens des traitements de texte) un texte fourni en ASCII, sur un nombre de colonnes donné, en découpant les mots selon les règles en usage pour le français.<br>On supposera que tous les caractères ont la même dimension (« fonte proportionnelle »).<br>Pour le découpage (éventuel) des mots (hyphenation), on prendra bien garde de distinguer les règles, qui seront stockées dans un fichier, du programme lui-même. On s'autorisera à utiliser un dictionnaire d'exceptions. | Pascal ou C | | | |
| | | Enrichissement de requêtes | Il s'agit de se placer dans la problématique suivante : on formule des requêtes à un moteur de recherche, les requêtes pouvant contenir un ou plusieurs mots, voire un syntagme ou une phrase. Sachant que les taux de réussite de telles requêtes ne sont pas toujours satisfaisants, on se propose d'enrichir ces requêtes (en ajoutant des mots, ou en utilisant les opérateurs classiques de moteurs de recherche...) pour améliorer le résultat.<br>Cet enrichissement sera réalisé sur la bases d'information de nature lexicale ou terminologique, stockées dans un réseau sémantique approprié.<br>Le but du projet n'est pas de réaliser un tel réseau sémantique à grande couverture, mais de spécifier, pour un domaine conceptuel fixé, les heuristiques à utiliser et l'architecture du réseau correspondant. Le projet donnera lieu à la réalisation d'une maquette, au sens industriel du terme. | Pascal ou C | | | |

| ANNEE | NIVEAU | SUJET | DESCRIPTIF | LANGAGE | RESPONSABLE | DIFFICULTE | GROUPE |
|---|---|---|---|---|---|---|---|
| | M1 | | | | | | |
| 1999-2000 | L3 | Correcteur orthographique (approche graphémique) | On se propose de réaliser un correcteur orthographique (lexical), qui, disposant d'un dictionnaire de formes fléchies, détecte les mots mal orthographiés, et propose si possible une correction.<br>Le programme prend un texte (ASCII brut) en entrée, et pour chaque forme non présente dans le dictionnaire, propose à l'utilisateur :<br>• de choisir un de ces remplaçants<br>o pour l'occurrence, ou pour toutes les occurrences<br>• de fournir un remplaçant<br>o pour l'occurrence, ou pour toutes les occurrences<br>o si le remplaçant n'est pas dans le lexique,<br>♣ de l'insérer dans le lexique pour ce texte,<br>♣ de l'insérer dans le lexique stable<br>• d'ignorer la correction, c'est-à-dire conserver le mot initial, pour cette occurrence ou pour toutes, en l'insérant ou non dans le lexique.<br>Les remplaçants proposés par le programme (premier cas plus haut) seront des mots figurant dans le dictionnaire, et ayant une certaine proximité avec le mot fautif.<br>Pour la recherche des mots proches, on appliquera diverses heuristiques, basées sur la forme du mot :<br>• repérage des bigrammes ou trigrammes impossibles en français<br>• redoublement ou dé-doublement de consonnes<br>• suppression/insertion de diacritiques<br>• insertion de lettres<br>• interversion de lettres<br>• etc.<br>Ces heuristiques, qui peuvent être plus nombreuses, seront étudiées linguistiquement afin de déterminer précisément leurs conditions d'application. | Pascal ou C | | | |
| | | Réaccentuation | Le programme est chargé de remettre les accents et autres signes diacritiques manquant dans un texte fourni en typographie dite pauvre.<br>On s'aidera d'un lexique de formes fléchies fourni. Bien sûr, certaines formes peuvent être réaccentuées directement, et d'autres sont ambigues. Une fois que l'algorithme principal sera établi, on envisagera des heuristiques pour lever les ambiguïtés. Exemple :<br>La ou le francais n'est pas accentue,<br>il y a de la gene,<br>mais quand le systeme m'accentue,<br>je suis moins gene!<br>On pourra s'inspirer du programme « Reacc » développé par le RALI à l'Université de Montréal. | Pascal ou C | | | |
| | | Correction d'étiquettes | Disposant d'un corpus étiqueté et lemmatisé (chaque mot du corpus est accompagné de son étiquette grammaticale et du lemme lui correspondant [cf. figure 1]), il s'agit d'écrire un programme permettant de corriger d'éventuelles erreurs d'étiquetage.<br>ce DETRMS ce<br>paragraphe SUBSMS paragraphe<br>résume VER3B résumer<br>les DETRMP les | Pascal ou C | | | |

| ANNEE | NIVEAU | SUJET | DESCRIPTIF | LANGAGE | RESPONSABLE | DIFFICULTE | GROUPE |
|---|---|---|---|---|---|---|---|
| | | | problèmes SUBMP problème<br>qui PRELMP qui<br>doivent VERB6 devoir<br>être AUXE être<br>résolus PPASMP résoudre<br>pour PREP pour<br>exploiter VINF exploiter<br>des PDES des<br>liaisons SUBSFP liaison<br>de PDEA de<br>télécommunicatio SUBSFS télécommunication<br>par PREP par<br>satellite SUBSMS satellite<br>FIG 1. Exemple de phrase du corpus<br>À ce programme, l'utilisateur fournira des règles qui auront la forme suivante :<br>Changer l'étiquette Y1 du mot X en Y2 si les étiquettes Z1 et Z2 sont présentes dans le contexte de X, le contexte étant défini comme l'ensemble des cinq mots précédant et des cinq mots suivant X.<br>Alors, le programme appliquera ces règles sur le corpus, en montrant à l'utilisateur les cas où la règle s'applique, de façon à ce qu'il puisse vérifier si la règle est pertinente.<br>Exemple : au vu de l'étiquetage fautif<br>l l DETRMS<br>' ' x<br>usager usager SUBSMS<br>désire désirer VERB3<br>établir établir VINF<br>des des PDES<br>liaisons liaison SUBSFP<br>on pourra envisager une règle : L'étiquette PDES (=Y1) du mot des (=X) devient DETRMP (=Y2) si VERB3 (=Z1) et SUBSFP (=Z2) se trouvent dans le contexte du mot.<br>On envisagera la possibilité d'appliquer plusieurs règles lors d'une passe, et on choisira de préférence de stocker ces règles dans un fichier. On notera que l'ordre d'application des règles est important.<br>À titre d'exemple, vous proposerez un ensemble de règles qui permette de corriger les erreurs d'étiquetage sur ``des''.<br>On pourra envisager d'autres formes de règles. | | | | |
| | | Calcul de collocations | On veut réaliser un programme de calcul des collocations dans un corpus étiqueté et lemmatisé.<br>Le but est de repérer et de compter les blocs de 2 ou 3 mots juxtaposés qui se retrouvent au moins deux fois dans le corpus. On s'intéressera aussi aux groupes « N à N », « N de N », etc., dont on calculera aussi les fréquences.<br>La sortie du programme consiste donc en une liste, classée par fréquence, de toutes les collocations qui y apparaissent au moins deux fois.<br>On attachera un soin particulier au choix de la structure de données, et à l'algorithme, qui doit faire le calcul en un temps raisonnable. | Pascal ou C | | | |

| ANNEE | NIVEAU | SUJET | DESCRIPTIF | LANGAGE | RESPONSABLE | DIFFICULTE | GROUPE |
|---|---|---|---|---|---|---|---|
| | | Extraction de sigles | L'objectif est d'être capable de construire, semi-automatiquement, la liste des sigles (acronymes) (par exemple EDF), et de leur forme développée (par exemple Electricité de France), utilisés dans un texte donné. Le programme aura deux fonctions :<br>1. Création (extraction) de la liste (triée, normalisée) des sigles du texte donné. Attention aux formes diverses d'un même sigle (Inalf, I.N.A.L.F, INaLF).<br>2. Proposition (interactive) de syntagmes candidats pour la forme développée, quand on en trouve dans le texte.<br>Le texte fourni est en ASCII brut. | Pascal ou C | | | |
| | | Justification et "hyphénation" | Il s'agit de réaliser un programme qui justifie (au sens des traitements de texte) un texte fourni en ASCII, sur un nombre de colonnes donné, en découpant les mots selon les règles en usage pour le français.<br>On supposera que tous les caractères ont la même dimension (« fonte proportionnelle »).<br>Pour le découpage (éventuel) des mots (hyphenation), on prendra bien garde de distinguer les règles, qui seront stockées dans un fichier, du programme lui-même. On s'autorisera à utiliser un dictionnaire d'exceptions. | Pascal ou C | | | |
| | | Concordancier | Réalisation d'un « concordancier » : disposant d'un corpus étiqueté et lemmatisé (chaque mot du corpus est accompagné de son étiquette grammaticale et du lemme lui correspondant (cf. figure 1), il s'agit d'extraire les phrases de ce corpus comportant des séquences d'étiquettes et/ou de lemmes données.<br>Les séquences<br>Les séquences sont des suites d'étiquettes et/ou de lemmes, et comportant éventuellement des symbole ayant une interprétation particulière (cf. expressions rationnelles) :<br>*<br>signifie zéro ou plusieurs occurrences de l'entité (étiquette grammaticale ou lemme) précédente,<br>+<br>signifie une ou plusieurs occurrences de l'entité précédente.<br>Voici quelques exemples de séquences accompagnées de leur interprétation :<br>SUBS ADJ un substantif, suivi d'un adjectif<br>SUBS ADJ* PREP un substantif, suivi d'un nombre quelconque d'adjectifs, suivi d'une préposition<br>SUBS ADJ+ PREP un substantif, suivi d'au moins un adjectif, suivi d'une préposition.<br>Exemple de phrase à extraire<br>Considérons la phrase de la figure 1. Cette phrase doit être extraite si la séquence recherchée est VINF des SUBS mais ne doit pas l'être si la séquence recherchée est SUBS ADJ des SUBS.<br>Extraction des phrases<br>Le programme propose à l'utilisateur de saisir une séquence (on rappelle que cette séquence peut comprendre des étiquettes et des lemmes), extrait les phrases qui comprennent cette séquence, et les stocke dans un fichier (on envisagera aussi leur visualisation). À titre d'exemple, si l'on s'intéresse au lemme des, on pourra demander au programme d'extraire du corpus les phrases correspondant aux séquences suivantes :<br>1. VINF des SUBS<br>2. PREP VINF des ADJ* SUBS<br>3. PREP des ADV* ADJ* SUBS<br>4. SUBS ADJ+ des ADV* ADJ* SUBS<br>Remarques<br>• les étiquettes grammaticales utilisées dans les séquences sont plus générales que celles utilisées dans le corpus. En particulier, aucune information de genre et de nombre n'est donnée dans ces séquences, | Pascal ou C | | | |

| ANNEE | NIVEAU | SUJET | DESCRIPTIF | LANGAGE | RESPONSABLE | DIFFICULTE | GROUPE |
|---|---|---|---|---|---|---|---|
| | | | • les prépositions ``à'' et ``de'' sont étiquetées PDEA dans le corpus sur lequel vous travaillerez mais doivent être considérées comme des prépositions. | | | | |
| | M1 | Repérage des "entités nommées" | On regroupe sous le terme « entités nommées » les noms de personnes, de lieux, les dates, etc. Le but du projet consiste à repérer automatiquement un sous-ensemble homogène parmi les entités (par exemple les noms de personnes ou les noms d'entreprises).<br><br>Le programme envisagé procède par étapes successives sur le même texte. Dans une première étape, grâce aux dictionnaires spécialisés (cf. plus loin), on étiquette les mots simples. On partira d'un corpus déjà étiqueté morpho-syntaxiquement. La seconde étape, basée sur des règles, permettra de regrouper ces mots simples pour former des entités nommées. Enfin, une troisième étape, facultative dans ce projet, pourrait tenter de relier entre elles les entités co-référentes. Les éléments seront marqués dans le texte au moyen d'un système de balisage de type SGML.<br><br>Le système repose donc, outre le programme lui-même, sur les ressources suivantes :<br>• des dictionnaires :<br>o dictionnaire de la langue générale pour repérer les mots inconnus.<br>o dictionnaires spécialisés de prénoms, de noms de sociétés connues, etc.<br>o dictionnaires spécialisés d'« amorces », mots ou groupes de mots qui peuvent marquer le début d'une entité nommée.<br>• une « grammaire » permettant de combiner des mots repérés de natures diverses (mots inconnus, prénoms, amorces...) pour en faire une entité (un nom de personne).<br>Les dictionnaires et la grammaire sont stockés dans des fichiers textes. On assure ainsi une séparation nette entre les données et les traitement (le programme).<br>Exemple<br>Soit le texte « M. Jack Lang a gagné la mairie. »<br>"M." est stocké dans un dictionnaire "d'amorces"<br>"Jack" est stocké dans un dictionnaire de prénoms<br>"Lang" est inconnu du système<br>La première étape permet d'obtenir :<br><TR_PERSON>M.</TR_PERSON> <PERSON>Jack</PERSON> <UFIRSTUNKNOWN>Lang</UFIRSTUNKNOWN> a gagné la mairie.<br>Une règle de la forme<br>TR_PERSON PERSON UFIRSTUNKNOWN ==> PERSON<br>va pouvoir être activée.<br>D'où le résultat de la seconde étape :<br><PERSON>M. Jack Lang</PERSON> a gagné la mairie.<br>Une troisième étape permettrait d'identifier toutes les occurrences de Jack Lang<br><PERSON id=1>M. Jack Lang</PERSON> a gagné la mairie.<br>Sous réserve<br>Je devrais pouvoir fournir une partie des dictionnaires à utiliser. Il est souhaitable que le système stocke dans un fichier les mots inconnus susceptibles d'être des entités, pour pouvoir améliorer rapidement et dynamiquement la couverture. | Lisp, Prolog, Pascal, C, C++ ou java | | | |
| | | Manipulation d'automates | On se propose de réaliser un programme, ou mieux, une bibliothèque de fonctions, qui permette de réaliser divers traitements sur des automates à nombre fini d'états. | Lisp, Prolog, Pascal, C, C++ | | | |

| ANNEE | NIVEAU | SUJET | DESCRIPTIF | LANGAGE | RESPONSABLE | DIFFICULTE | GROUPE |
|---|---|---|---|---|---|---|---|
| | | | On peut envisager une interface graphique (pour la sortie et même pour l'entrée des automates), mais on pourra aussi se contenter de travailler sur des tables de transition en Ascii, dans une notation à la PC-Kimmo. (Il faudra bien sûr envisager une structure de données interne pour travailler sur les automates.)<br>Parmi les opérations que l'on devra pouvoir réaliser avec le système, il y aura au moins :<br>• Repérage de la catégorie de l'automate : complet ou non, déterministe ou non, avec ou sans epsilon-transitions...<br>• Complétion d'un automate<br>• Déterminisation d'un automate<br>• Minimisation d'un automate<br>• Suppression des epsilon-transitions d'un automate<br>Facultatif :<br>• Traduction d'une expression rationnelle en automate<br>• Traduction d'un automate en expression rationnelle | ou java | | | |
| | | Réseaux sémantiques | À partir de définitions provenant de dictionnaires électroniques, préalablement étiquetées, il s'agit de construire un « réseau sémantique ».<br>Pour chaque entrée (qui peut correspondre à un des sens d'une lexie), on repèrera les mots pleins qui participent à sa définition (il faut donc (1) distinguer les mots pleins des mots « outils », (2) repérer la définition proprement dite parmi l'ensemble des informations associées à une entrée (catégorie, exemples, synonymes...)), et on construira un réseau reliant l'entrée à tous ces mots pleins.<br>Ce réseau pourra être stocké sous forme de fichier Ascii, dans un format du genre de celui de WordNet.<br>On pourra aussi, éventuellement, proposer une interface graphique permettant de visualiser graphiquement le réseau, voire de le modifier.<br>Dans un deuxième temps, on réalisera un programme exploitant ce réseau pour désambiguïser les sens d'un mot en contexte. Principe : étant donnée une phrase contenant le mot concerné, on repère les mots pleins qui l'entourent (contexte), et on recherche ces mots dans le réseau. Alors le sens correspondant est celui qui est le plus proche (dans un sens qu'il faut précisément définir) des mots pleins de son contexte. | Lisp, Prolog, Pascal, C, C++ ou java | | | |
| | | Génération de texte en LFG (lisp) | Le programme doit pouvoir afficher la ou les phrases correspondant à une structure fonctionnelle supposée bien formée (cohérente et complète). Le programmeur fournira une petite grammaire LFG pour la langue de son choix.<br>1. Première étape : Écrire un programme qui génère le langage correspondant à une grammaire CF quelconque. Le programme pourra « boguer » en cas de récursion gauche. On pourra produire par exemple un réseau d'automates.<br>2. Seconde étape : Compléter ce programme pour tenir compte des équations fonctionnelles. Seul le langage correspondant à la structure fonctionnelle devra être produit. Procéder par étapes :<br>o analyser les équations du type v = ( ^ obj )<br>o ensuite les equations du type v = (^ (v pcas )-obj)<br>o ensuite les équation contraintes<br>o ensuite les equations existentielles<br>3. Troisième étape : Retrouver les formes fléchies dans un dictionnaire à partir des lemmes et informations morphologiques des mots. | Lisp, Prolog, Pascal, C, C++ ou java | | | |
| | | "Complétion" automatique | On se propose de réaliser un programme qui accélère la vitesse de frappe d'un utilisateur en proposant - de façon intelligente - la fin des mots en cours de frappe.<br>Un tel système, pour être vraiment performant, devrait faire de la prédiction syntaxique en temps réel. On se contentera dans ce projet de faire un système qui propose la suite la plus probable des mots. | Emacs Lisp | | | |

| ANNEE | NIVEAU | SUJET | DESCRIPTIF | LANGAGE | RESPONSABLE | DIFFICULTE | GROUPE |
|---|---|---|---|---|---|---|---|
| | | | La méthode repose sur un calcul de probabilité basé sur la fréquence d'apparition de chaque mot dans un corpus préalablement fourni. On construit un graphe dont chaque chemin correspond à un mot (une lettre par arc), et dont chaque arc porte une probabilité.<br>Par exemple, la probabilité de la prochaine lettre de la suite ``v-o-l-u'' est respectivement de 0/9, 8/9 (3+5/9), 1/9, 0/9 pour les lettres `b', `m', `p' et `t' si nous avons dans le corpus :<br>0 occurrences du mot volubile<br>3 '' volume<br>5 '' volumineux<br>1 '' voluptueux<br>0 '' volute<br>La lettre la plus probable, ici ``m'', sera proposée à l'utilisateur, et un nouveau calcul est effectué pour la suite de lettres ``v-o-l-u-m''.<br>Un soin particulier sera apporté à l'interface de ce programme : l'utilisateur doit pouvoir accepter ou revenir facilement sur les propositions du système, etc. Un langage particulièrement adapté au projet serait Emacs Lisp, ce qui permettrait d'intégrer la complétion à l'éditeur Emacs. | | | | |
| | | Désambiguïsation par automates | Partant d'un texte dont chaque mot est étiqueté avec toutes ses catégories morpho-syntaxiques possibles, on va s'attacher à lever certaines ambiguïtés non pertinentes, c'est-à-dire à supprimer certaines des étiquettes attachées à certains mots.<br>Pour mener à bien cette tâche, on extraiera du corpus donné les éléments pertinents, ce qui permettra d'établir des « règles contextuelles », que l'on appliquera ensuite au corpus.<br>1. Étude linguistique<br>On s'intéressera à une catégorie donnée, par exemple « verbe conjugué ». On extraira du texte une première liste de mots qui peuvent être des verbes conjugués (ceux dont la liste d'étiquettes comprend la catégorie V), puis, après désambiguïsation, une deuxième liste comprenant seulement les mots qui sont effectivement des verbes conjugués dans le texte.<br>On relèvera alors les contextes locaux (catégories précédentes, catégories suivantes) qui permettent de choisir la catégorie retenue pour un mot donné en français.<br>Par exemple, si X est marqué N ou V, et si X est précédé d'un Det, alors X doit être marqué N (l'étiquette V doit être ôtée).<br>Etablir ainsi des séquences de 2 ou 3 catégories impossibles en français.<br>On aura intérêt à subdiviser certaines catégories du DELAF, par exemple mettre à part parmi les pronoms, les formes clitiques (je, tu, il(s), elle(s), nous, vous, se, on, ce, y, en, le, la, les, lui, leur).<br>Attention au cas où les mots suivants ou précédents ont plusieurs catégories possibles.<br>2. Programmation et implémentation<br>Une fois les règles dégagées, on réalisera un programme qui procédera à la désambiguisation du corpus en appliquant les règles. On conseille d'implémenter les règles sous forme de transducteur.<br>On fournit deux textes étiquetés (d'une dizaine de pages) à l'aide du DELAF, qui serviront de base à l'élaboration des règles de désambiguisation.<br>Notations du DELAF :<br>Général Pour les verbes<br>V verbe<br>N nom<br>DET déterminant (article)<br>DETP déterminant possessif<br>DETQ déterminant numéral | Lisp, Prolog, Pascal, C, C++ ou java | | | |

| ANNEE | NIVEAU | SUJET | DESCRIPTIF | LANGAGE | RESPONSABLE | DIFFICULTE | GROUPE |
|---|---|---|---|---|---|---|---|
| | | | INTJ interjection<br>CNJS conjonction de subordination<br>CNJC conjonction de coordination<br>A adjectif<br>PRON pronom<br>ADV adverbe<br>Xin préfixe<br>P présent indicatif<br>S subjonctif présent<br>Y impératif<br>K participe passé<br>F futur<br>W infinitif<br>I imparfait<br>C conditionnel<br>G participe présent<br>T subjonctif imparfait<br>Inv invariable<br>1, 2 ou 3 : personne<br>m, f, s, p : genre, nombre<br>Exemple<br>Forme générale : mot lemme cat:morpho ...<br>vient venir V33:P3s<br>livre livre N1:ms livre N21:fs livrer V3:P1s:P3s:S1s:S3s:Y2s | | | | |
| | | Compression de texte | On doit réaliser un couple de programmes qui permette de compresser-décompresser un texte.<br>Le principe de l'algorithme de compression est le suivant : sur des bases statistiques, en fonction du texte en entrée, on détermine des chaînes les plus longues possible (en particulier préfixes et suffixes) présentes avec une fréquence raisonnable, et on remplace dans le texte toutes les occurrences de ces chaînes par des non-mots -ou des mots absents du texte- (p. ex. « aa »), plus courts.<br>Bien sûr, le fichier compressé contiendra aussi une table de conversion, qui permettra au programme de décompression de reconstituer le fichier original. | Lisp, Prolog, Pascal, C, C++ ou java | | | |
| 1998-1999 | L3 | | | | | | |
| | M1 | | | | | | |
| 1997-1998 | L3 | Extraction de phrases | | Pascal ou C | | | |
| | | Correcteur ortaugrafique | On veut réaliser un correcteur orthographique (lexical) qui fait des propositions de correction lorsqu'il rencontre un mot n'appartenant à son lexique.<br>Fonctionnement : on dispose d'un lexique de formes fléchies. Un texte est fourni en entrée du programme (ascii brut, sans marquage). Pour chaque mot du texte non présent dans le lexique, l'algorithme déterminera un ou plusieurs mots proches (mais figurant dans le lexique), et proposera à l'utilisateur : | Pascal ou C | | | |

| ANNEE | NIVEAU | SUJET | DESCRIPTIF | LANGAGE | RESPONSABLE | DIFFICULTE | GROUPE |
|---|---|---|---|---|---|---|---|
| | | | de choisir un de ces remplaçants<br>pour l'occurrence, ou pour toutes les occurrences<br>de fournir un remplaçant<br>pour l'occurrence, ou pour toutes les occurrences<br>si le remplaçant n'est pas dans le lexique,<br>de l'insérer dans le lexique pour ce texte,<br>de l'insérer dans le lexique stable<br>d'ignorer la correction, c'est-à-dire conserver le mot initial, pour cette occurrence ou pour toutes, en l'insérant ou non dans le lexique.<br>En qui concerne la détermination de la proximité entre deux mots, on réalisera un « panachage » entre les deux méthodes classiques :<br>approche graphémique : règles basées sur la forme : redoublement de consonnes, substitutions de voyelles, interversion de lettres (voisines sur le clavier, p.ex), etc.<br>approche phonétique : règles basée sur la proximité phonétique : substitution de `k' par `qu', `c' (si suivi de la bonne voyelle), de `ph' par `f', gestion des `e' muets, etc.<br>Remarques<br>Il n'est pas raisonnable de charger tout le dictionnaire en mémoire. Il faut envisager des chargements partiels.<br>On peut envisager un dictionnaire parallèle de formes phonétiques associées, ou une table de correspondance graphème-phonème. Dans les deux cas, ce n'est pas incompatible avec le développement d'heuristiques spécifiques.<br>On peut limiter la complexité des calculs en tenant compte des bigrammes possibles et impossibles en français. | | | | |
| | | Correction d'étiquettes | | Pascal ou C | | | |
| | | Prédiction d'étiquettes | Disposant d'un lexique de formes fléchies, associées à une étiquette (partie du discours, + informations morpho-syntaxiques), et d'un texte à étiqueter, le programme doit étiqueter les formes reconnues, et tenter de deviner, uniquement sur des bases morphologiques, l'étiquette probable des formes qui ne sont pas dans le lexique.<br>On se servira d'une table classique morphologique (grammaire Becherelle), la terminaison du mot appartient à l'une des terminaisons décrites, on en prédit l'étiquette.<br>Exemple : la terminaison de vermifugerions "erions" donne à croire que le mot est un verbe au conditionnel présent première personne du singulier. | Pascal ou C | | | |
| | | Extraction de sigles | L'objectif est d'être capable de construire, semi-automatiquement, la liste des sigles (acronymes), et de leur forme développée, utilisés dans un texte donné. Le programme aura trois fonctions :<br>Création (extraction) de la liste (triée, normalisée) des sigles du texte donné.<br>Correction (automatique) si les sigles apparaissent sous plusieurs formes dans le texte (I.N.A.L.F, Inalf, INaLF).<br>Proposition (interactive) de syntagmes candidats pour la forme développée, quand on en trouve dans le texte.<br>Le texte fourni est en ASCII brut. | Pascal ou C | | | |
| | M1 | Complétion automatique | | Lisp, Prolog, Pascal, C, C++ ou java | | | |

| ANNEE | NIVEAU | SUJET | DESCRIPTIF | LANGAGE | RESPONSABLE | DIFFICULTE | GROUPE |
|---|---|---|---|---|---|---|---|
| | | Désambiguïsation par automates | | Lisp, Prolog, Pascal, C, C++ ou java | | | |
| | | Traduction de nombres | | Lisp, Prolog, Pascal, C, C++ ou java | | | |
| | | Compréhension de dates | | Lisp, Prolog, Pascal, C, C++ ou java | | | |
| | | Analyseur syntaxique | | Lisp, Prolog, Pascal, C, C++ ou java | | | |
| 1996-1997 | L3 | | Écriture d'un programme permettant :<br>1. d'extraire des phrases correspondant à des séquences données,<br>2. de corriger des étiquettes grammaticales à partir de règles à définir.<br>1. Extraction de phrases correspondant à des séquences données<br>Disposant d'un corpus étiqueté et lemmatisé (chaque mot du corpus est accompagné de son étiquette grammaticale et du lemme lui correspondant), il s'agit, dans un premier temps, d'extraire les phrases de ce corpus comportant des séquences d'étiquettes et/ou de lemmes données.<br>1.1. Les séquences<br>Les séquences représentent des suites admissibles d'étiquettes grammaticales et/ou de lemmes que l'on peut représenter sous forme d'expressions régulières.<br>Deux symboles ont une signification particulière dans les séquences:<br>*<br>signifie zéro ou plusieurs occurrences de l'entité (étiquette grammaticale ou lemme) précédente,<br>+<br>signifie une ou plusieurs occurrences de l'entité précédente.<br>Voici quelques exemples de séquences accompagnées de leur interprétation :<br>SUBS ADJ un substantif, suivi d'un adjectif<br>SUBS ADJ* PREP un substantif, suivi d'un nombre quelconque d'adjectifs, suivi d'une préposition<br>SUBS ADJ+ PREP un substantif, suivi d'au moins un adjectif, suivi d'une préposition.<br>1.2. Exemple de phrase à extraire<br>Considérons la phrase suivante extraite du corpus :<br>ce DETRMS ce<br>paragraphe SUBSMS paragraphe<br>résume VER3B résumer<br>les DETRMP les<br>problèmes SUBMP problème<br>qui PRELMP qui<br>doivent VERB6 devoir<br>être AUXE être | Pascal | | | |

| ANNEE | NIVEAU | SUJET | DESCRIPTIF | LANGAGE | RESPONSABLE | DIFFICULTE | GROUPE |
|---|---|---|---|---|---|---|---|
| | | | résolus PPASMP résoudre<br>pour PREP pour<br>exploiter VINF exploiter<br>des PDES des<br>liaisons SUBSFP liaison<br>de PDEA de<br>télécommunicatio SUBSFS télécommunication<br>par PREP par<br>satellite SUBSMS satellite<br>Cette phrase doit être extraite si la séquence recherchée est VINF des SUBS mais ne doit pas l'être si la séquence recherchée est SUBS ADJ des SUBS.<br>1.3. Extraction des phrases<br>Il vous est ici demandé d'extraire du corpus les phrases correspondant aux séquences suivantes :<br>1. VINF des SUBS<br>2. PREP VINF des ADJ* SUBS<br>3. PREP des ADV* ADJ* SUBS<br>4. SUBS ADJ+ des ADV* ADJ* SUBS<br>A chaque séquence sera associé un fichier dans lequel vous rangerez les phrases correspondant à la séquence.<br>Remarques<br>• les étiquettes grammaticales utilisées dans les séquences sont plus générales que celles utilisées dans le corpus. En particulier, aucune information de genre et de nombre n'est donnée dans ces séquences,<br>• les prépositions ``à'' et ``de'' sont étiquetées PDEA dans le corpus sur lequel vous travaillerez mais doivent être considérées comme des prépositions.<br>Indication Il est possible de représenter les séquences sous forme de tableau où chaque élément correspond à une entité. La séquence SUBS ADJ+ PREP peut ainsi être représentée par le tableau suivant, dans lequel l'élément ``+'' indique qu'on peut répéter l'élément précédent. Une séquence est reconnue dans une phrase lorsqu'on a réussi à parcourir tout le tableau en lisant la phrase.<br>SUBS ADJ + PREP<br>2. Correction des étiquettes grammaticales<br>Il s'agit ici d'écrire des règles et un programme permettant de corriger les éventuelles erreurs d'étiquetage que vous aurez pu relever sur le mot ``des''.<br>Les règles auront la forme suivante :<br>Changer l'étiquette Yl du mot X en Y2 si les étiquettes Zl et Z2 sont présentes dans le contexte de X, le contexte étant défini comme l'ensemble des cinq mots précédant et des cinq mots suivant X.<br>Vous pourrez ajouter des contraintes sur ces règles (comme par exemple le fait que Zl et Z2 doivent se suivre).<br>Il vous faudra vérifier que les règles que vous écrivez ne génèrent pas plus d'erreurs qu'elles n'en corrigent.<br>3. Extension<br>Vous pourrez definir de nouvelles séquences associées à ``des'' si vous le jugez utile (ou si vous avez terminé le projet en avance). Vous pourrez également vous intéresser à d'autres mots que ``des''. | | | | |
| | | | Écriture d'un programme permettant : | Pascal | | | |

| ANNEE | NIVEAU | SUJET | DESCRIPTIF | LANGAGE | RESPONSABLE | DIFFICULTE | GROUPE |
|---|---|---|---|---|---|---|---|
| | | | 1. d'extraire des pseudo-affixes en vue de sélectionner les séquences à abréger,<br>2. de générer automatiquement des codes correspondant aux séquences retenues,<br>3. de prendre en compte ces codes pour simplifier l'écriture d'un mot.<br>1. Extraction de pseudo-affixes<br>Il s'agit ici de trouver, à partir des mots présents dans un corpus, les séquences de caractères les plus représentées. Seules les séquences supérieures à n (à choisir et/ou déterminer) caractères seront considérées. Il faudra classer les séquences obtenues par ordre décroissant du nombre d'occurrences.<br>L'extraction reposera sur l'algorithme présenté dans l'article de Nagao et Mori ci-joint. Vous commencerez par vous intéresser aux séquences finales, qu'on pourrait qualifier de pseudo-suffixes. Le corpus sur lequel vous travaillerez est un corpus étiqueté et lemmatisé identique à celui décrit dans le projet 1.<br>Une fois les séquences extraites, vous devrez en choisir un certain nombre sur lesquelles vous travaillerez, ce choix devant être motivé.<br>Remarque Il vous faudra choisir entre un travail sur formes fléchies et sur lemmes pour extraire vos séquences. Vous pouvez choisir l'un ou l'autre (les deux présentent un intérêt), mais vous devez indiquer les raisons de votre choix.<br>2. Génération automatique de codes<br>A partir des séquences retenues, il vous est ici demandé de générer automatiquement un code pour chacune d'elles. Ce code prendra la forme d'une suite de ``a'' et de ``b''. La séquence ``ment'' peut par exemple être représentée par la suite ``aa'', de telle sorte que pour signifier ``changement'' on peut se contenter de taper ``changeaa''. Le gain en lettres est ici faible, mais il faut tenir compte du fait que le code généré doit être non ambigu. Ainsi, si l'on avait choisi ``a'' pour représenter la séquence ``ment'', on n'aurait pu décider si ``changea'' représentait ``changement'' ou le passé simple du verbe ``changer''.<br>Vous pourrez vous inspirer de l'algorithme de Huffman pour générer les codes. Je vous fournirai cet algorithme en temps utile.<br>3. Prise en compte des codes<br>Cette partie concerne le remplacement automatique des codes par les séquences associées. Il s'agit ici d'écrire une fonction lisant un mot, fourni par l'utilisateur, éventuellement incomplet mais comportant un code et renvoyant le mot complet. Suivant le code généré précédemment, plusieurs stratégies peuvent être utilisées. Si le code reste ambigu, on peut demander à l'utilisateur de séparer le mot incomplet du code par un caractère de contrôle (``?'', par exemple, qui n'est attesté à l'intérieur d'aucun mot de la langue française). Si le code généré est non ambigu, un tel caractère est superflu.<br>4. Extension<br>On peut envisager le même traitement sur les séquences initiales des mots (pseudo-préfixes), voire sur des séquences quelconques. La non ambiguïté du code devient alors un critère de première importance. Essayer d'envisager un tel traitement. | | | | |
| | M1 | Traduction de nombres écrits en lettres | 1. Etude linguistique<br>Etablir vocabulaire et grammaire (catégories et graphe de transition) pour les nombres dans chacune des langues choisies.<br>2. Programme et implémentation<br>Implémenter un automate (par exemple sous forme de transducteur) qui reconnaisse si la séquence (entrée par l'utilisateur) est bien un nombre dans la langue source, et la traduise en chiffre.<br>Exemples : mille deux cents -> 1200<br>un mille ->erreur<br>mille un -> 1001<br>Implementer un automate qui génère toutes les séquences possibles dans la langue cible. | Emacs Lisp ou Prolog | | | |

| ANNEE | NIVEAU | SUJET | DESCRIPTIF | LANGAGE | RESPONSABLE | DIFFICULTE | GROUPE |
|---|---|---|---|---|---|---|---|
| | | | Exemples : 1200 -> one thousand (and) two hundred, a thousand (and) two hundred<br>mille un -> one thousand and one, a thousand and one | | | | |
| | | Compréhension de dates écrites en lettres | 1. Etude linguistique<br>Etablir vocabulaire et grammaire (catégories et graphe de transition) pour les dates dans la langue choisie.<br>2. Programmation et implémentation<br>Soit analyser la séquence linguistique (entrée au clavier par l'utilisateur), vérifier que c'est bien une date et la traduire en chiffre. Attention aux expressions elliptiques, et aux déictiques.<br>Exemples : le dix-huit février -> 18/02/1997<br>le décembre 18 -> ceci n'est pas une date<br>demain -> 23/02/1995 (appel horloge système)<br>le lundi 14 décembre 36 -> 14/12/1936<br>en décembre 18 -> 01/12/1918 < d < 31/12/1918<br>Soit générer toutes les séquences linguistiques a partir d'une date en chiffres (entrée au clavier par l'utilisateur) :<br>Exemples : 18/02/95 -> le dix-huit février mille-neuf-cent-quatre-vingt-quinze, le 18 février 1995, etc. (Calculs supplementaires pour : le samedi 18 février, le troisième samedi du mois courant...)<br>23/02/95 -> le jeudi 23 février, demain (appel horloge système)...<br>29/02/95 -> 1995 n'est pas une année bissextile<br>32/06/97 -> Ceci n'est pas une date | Emacs Lisp ou Prolog | | | |
| | | Désambiguation par automates et extraction de catégories à partir d'un texte étiqueté | Il s'agit, à partir d'un texte découpé en mots, chacun étiqueté avec toutes ses catégories grammaticales possibles, de lever certaines ambiguïtés non pertinentes.<br>On s'attachera à l'extraction d'une catégorie donnée, par exemple (( Verbe conjugué )). On extraira du texte une première liste de mots qui peuvent être des verbes conjugués, puis, après désambiguation, une deuxième liste comprenant seulement les mots qui sont effectivement des verbes conjugués dans le texte.<br>Il est conseille de procéder comme suit :<br>1. Etude linguistique<br>Relever les contextes locaux (catégories précédentes, catégories suivantes) qui permettent de choisir la catégorie retenue pour un mot donné en francais.<br>Par exemple, si X est marqué N ou V, Det + X => X = N.<br>Etablir ainsi des séquences de 2 ou 3 catégories impossibles en francais<br>On aura interêt à subdiviser certaines catégories du DELAF, par exemple mettre à part parmi les pronoms, les formes clitiques (je, tu, il(s), elle(s), nous, vous, se, on, ce, y, en, le, la, les,lui, leur ).<br>Attention au cas ou les mots suivants ou précédents ont plusieurs catégories possibles.<br>2. Programmation et implementation<br>On conseille d'implémenter les règles dégagées sous forme de transducteurs.<br>On fournit deux textes étiquetés (d'une dizaine de pages) à l'aide du DELAF, qui serviront de base à l'élaboration des règles de désambiguïsation. Le programme devra être testé sur ces deux textes, ainsi que sur un troisième qui n'aura pas été étudié a priori.<br>Rappel des notations du DELAF :<br>V : verbe<br>N : nom | C ou Pascal | | | |

| ANNEE | NIVEAU | SUJET | DESCRIPTIF | LANGAGE | RESPONSABLE | DIFFICULTE | GROUPE |
|---|---|---|---|---|---|---|---|
| | | | DET : déterminant (article)<br>DETP : déterminant possessif<br>DETQ : déterminant numéral<br>INTJ : interjection<br>CNJS : conjonction de subordination<br>CNJC : conjonction de coordination<br>A : adjectif<br>PRON : pronom<br>ADV : adverbe<br>Xin : préfixe<br>Pour les verbes :<br>P : présent indicatif<br>S : subjonctif présent<br>Y : imperatif<br>K : participe passé<br>F : futur<br>W : infinitif<br>I : imparfait<br>C : conditionnel<br>G : participe présent<br>T : subjonctif imparfait<br>Inv : invariable<br>1, 2 ou 3 : personne<br>m, f, s, p : genre, nombre<br>Exemple Forme générale: mot lemme cat:morpho ...<br>vient venir V33:P3s<br>livre livre N1:ms livre N21:fs livrer V3:P1s:P3s:S1s:S3s:Y2s | | | | |
| | | | | Emacs Lisp ou Prolog | | | |